Proceedings of the 19th Sound and Music Computing Conference

Romain Michon, Laurent Pottier, Yann Orlarey

▶ **To cite this version:**

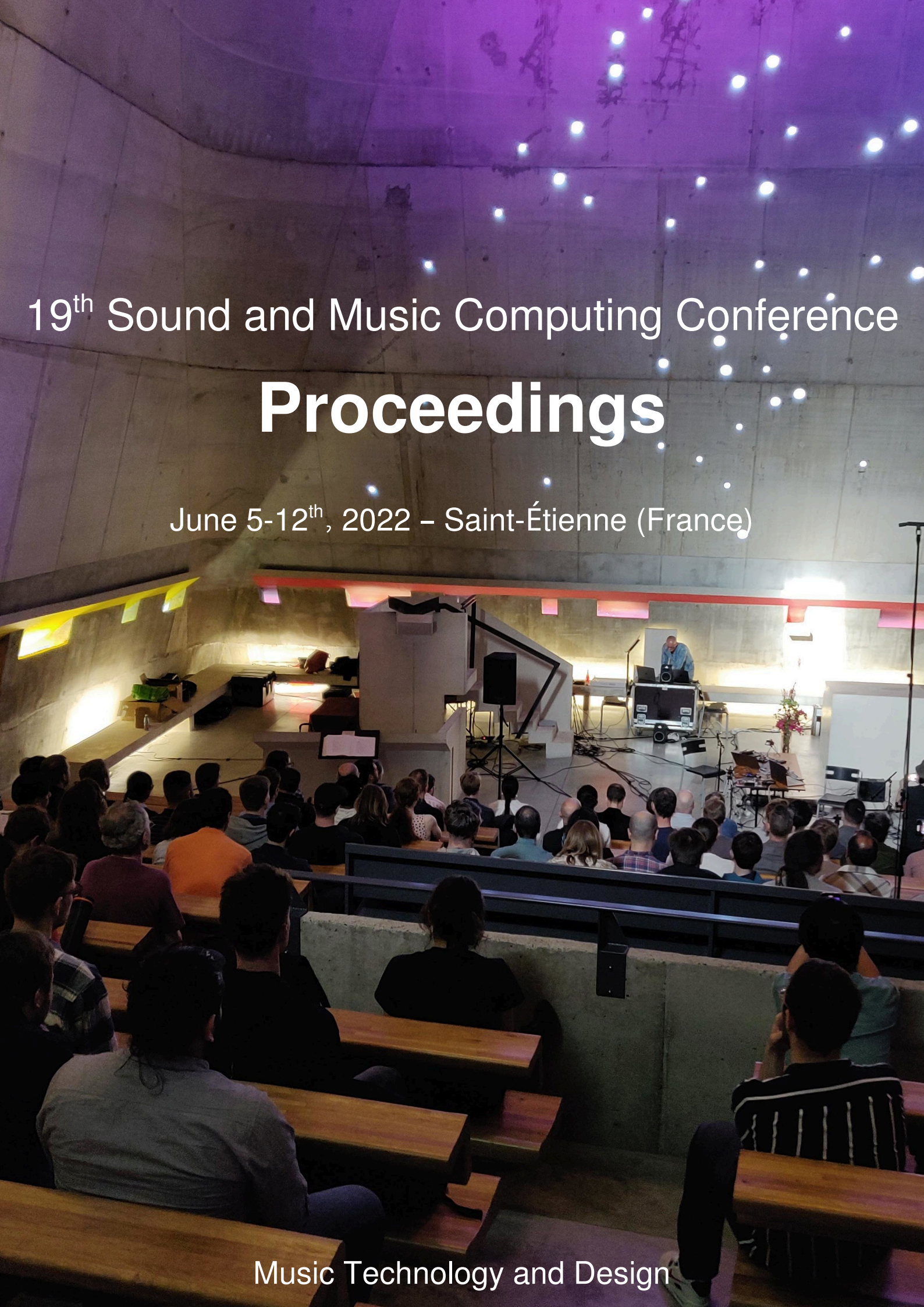
Romain Michon, Laurent Pottier, Yann Orlarey. Proceedings of the 19th Sound and Music Computing Conference. SMC Network, 2022, 978-2-9584126-0-9. hal-03927188

HAL Id: hal-03927188
https://inria.hal.science/hal-03927188

Submitted on 9 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



19th Sound and Music Computing Conference

Proceedings

June 5-12th, 2022 – Saint-Étienne (France)

Music Technology and Design

SMC 2022
St-Etienne
France

SMC/JIM/IFC 2022

Proceedings of the 19th Sound and Music Computing Conference

June 5-12th, 2022

Romain Michon, Laurent Pottier, and Yann Orlarey, eds.

19th Sound and Music Computing Conference
June 5-12th, 2022

SMC-22 is organized by:

With the support of:

Proceedings of the 19th Sound and Music Computing Conference
Romain Michon, Laurent Pottier, and Yann Orlarey, Editors
ISBN: 978-2-9584126-0-9
EAN: 9782958412609
Website: `https://smc22.grame.fr`

Bibtex:

```
@proceedings{19SMCConf,
Editor = {Romain Michon, Laurent Pottier and Yann Orlarey},
Organization = {Université Jean Monnet, INRIA, and GRAME-CNCM},
Publisher = {SMC Network},
Title = {Proceedings of the 19th Sound and Music Computing Conference},
Year = {2022}}
```

These proceedings, and all the papers included in it, are an open-access publication distributed under the terms of the Creative Commons Attribution 4.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and source are credited. To view a copy of this license, visit `http://creativecommons.org/licenses/by/4.0/` or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

Typeseting by Romain Michon using LaTeX.

Table of Contents

Organizational Committee and Technical Team **ix**

Scientific Committee **x**

Artistic Committee **xii**

Artistic Program **xiv**

SMC-22 Preface **xvi**

IFC-22 Preface **xviii**

Keynotes **xxi**

Presentation of Lightning and Industry Talks **xxiii**

SMC-22 Paper Session 1 **1**

Emulating Diode Circuits with Differentiable Wave Digital Filters – Jatin Chowdhury and Christopher Johann Clarke . . . 2

Approaching Spatial Audio Processing by Means of Decorrelation and Ring Modulation in Ambisonics – Paul Goutmann and Alain Bonardi . . . 10

Modular Physical Models in a Real-Time Interactive Application – Silvin Willemsen, Titas Lasickas, and Stefania Serafin . . . 17

Rubbing a Physics Based Synthesis Model: From Mouse Control to Frictional Haptic Feedback – Marius George Onofrei, Federico Fontana, and Stefania Serafin . . . 25

'1E0A': a Computational Approach to Rhythm Training – Noel Alben and Ranjani Hg . . . 33

Delia: A Noise-Based Virtual Synthesizer – Cecilia Rossi, Giorgio Presti, and Federico Avanzini . . . 41

For Lisa. A Piano-Based Sonification Project of Gravitational Waves – Andrea Valle and Valerya Korol . . 46

Sonifying an Office Gadget to Indicate Air Quality – Rod Selfridge, Carlo Barone, and Sandra Pauletto . . 54

A Perceptual Study of Sound Ecology in Peripheral Sonification – Maxime Poret, Catherine Semal, and Myriam Desainte-Catherine . . . 62

Speleoacoustics in Southern Ardèche for Auralizations and Music Experimentation – Luna Valentin, Miriam Kolar, and Philippe Monteil . . . 70

SMC-22 Paper Session 2 **78**

How Live is Live Coding? The case of Tidal's Longest Night – Raphaël Forment and Alex McLean . . . 79

Detection of Factors Affecting Cognitive Function in Environmental Sounds – Yasuhiro Kawahara, Hiroshi Yoshida, Laurent Pottier, and Pierre Maret . . . 87

Sound Design Ideation: Comparing Four Sound Designers' Approaches – Rod Selfridge and Sandra Pauletto 92

Spatial Audio Mixing in Virtual Reality – Simon Rostami Mosen, Anders Riddersholm Bargum, Oddur Ingí Kristjánsson, and Stefania Serafin . . . 100

Issues of Ubimus Archaeology: Reconstructing Risset's Music – Victor Lazzarini, Damián Keller, and Nemanja Radivojević . . . 107

Transformer and LSTM Models for Automatic Counterpoint Generation using Raw Audio – Lars Bentsen, Riccardo Simionato, Michael Joseph Krzyzaniak, and Benedikte Wallace . . . 115

Deep Embeddings for Robust User-Based Amateur Vocal Percussion Transcription – Alejandro Delgado, Emir Demirel, Vinod Subramanian, Charalampos Saitis, and Mark Sandler . . . 123

A Data-Driven Methodology for Considering Feasibility and Pairwise Likelihood in Deep Learning Based Guitar Tablature Transcription Systems – Frank Cwitkowitz, Jonathan Driedger, and Zhiyao Duan . 131

Muco: A Music Computing Learning Application – Patricia Hu, Oliver Hödl, Peter Reichl, Fares Kayali, Iris Eibensteiner, Bernhard Taufner, Sigrid Schefer-Wenzl, and Igor Miladinovic . . . 139

Improving Chord Prediction in Jazz Music using Melody Information – Leonhard Driever, Mels Loe Jagt, Kuan Lon Vu, Daniel Harasim, Andrew McLeod, and Martin Rohrmeier . . . 147

Speleoacoustics in Southern Ardèche for Auralizations and Music Experimentation – Luna Valentin, Miriam Kolar, and Philippe Monteil . . . 155

SMC-22 Paper Session 3 **162**

Kinesthesis: Designing of an Interactive Voice Software for Theatre Performance – Tilemachos Moussas, Natalia Kotsani, and Anastasia Georgaki . . . 163

FPGA-accelerated Real-Time Audio in Pure Data – Clemens Wegener, Max Neupert, and Sebastian Stang 170

Designing Interactive Sonifications For the Exploration of Dance Phrase Edges – Andreas Bergsland . . . 178

Effect of Using Individually Simulated HRTFs On the Outcome of Tournament Selection Procedures In a Virtual Environment – Michael Oehler, Minh Voong, Marlon Regener, and Maurício do V. M. da Costa 186
A Machine Learning Approach for MIDI to Guitar Tablature Conversion – Maximos Kaliakatsos-Papakostas, Grigoris Bastas, Dimos Makris, Dorien Herremans, Vassilis Katsouros, and Petros Maragos 194
Deep Convolutional Oscillator: Synthesizing Waveforms From Timbral Descriptions – Gordan Kreković . 202
Learning Sparse Analytic Filters for Piano Transcription – Frank Cwitkowitz, Mojtaba Heydari, and Zhiyao Duan 209
Redesigning a Piano Roll: A Melody Input Interface That Can Play Microtones With an Arbitrary Number of Keys – Tatsunori Hirai 217
A Comparison of Pitch Chroma Extraction Algorithms – Miguel Pérez Fernández, Holger Kirchhoff, and Xavier Serra Casals 224
A Robotic Drummer with a Flexible Joint: the Effect of Passive Impedance on Drumming – Seyed Mojtaba Karbasi, Alexander Refsum Jensenius, Rolf Inge Godøy, and Jim Torresen 232
Constructive Accumulation: a Look at Self-Organizing Strategies for Aggregating Temporal Events Along the Rhythm-Timbre Continuum – Nolan Lem 238

SMC-22 Paper Session 4 **246**
Algebraic Framework for Design and Qualitative Evaluation of Interactive Musical Systems Temporality – Myriam Desainte-Catherine, Irène Durand, Jean Haury, Bernard Serpette, and Sylviane Schwer . . 247
JS Fake Chorales: a Synthetic Dataset of Polyphonic Music with Human Annotation – Omar Peracha . . . 255
Sound-Accompanying Movements Enhance Perceived Arousal in Music Performance Even From a Distance – Hanna Järveläinen 263
Replicating Human Sound Localization with a Multilayer Perceptron – Eric Michael Sumner, Runar Unnthorsson, and Morris Riedel 271
Comparing Musical and Non-Musical Sonification Strategies for Auditory Guidance – Prithvi Ravi Kantan 279
Similarity of Nearest-Neighbor Query Results in Deep Latent Spaces – Philip Tovstogan, Xavier Serra, and Dmitry Bogdanov 287
Insights Into Transfer Learning Between Image and Audio Music Transcription – María Alfaro-Contreras, Jose J. Valero-Mas, Jose M. Iñesta, and Jorge Calvo-Zaragoza 295
SpecSinGAN: Sound Effect Variation Synthesis Using Single-Image GANs – Adrián Barahona-Ríos and Tom Collins 302
Quantum Simulation for Sound Synthesis: QHOSYN – Rodney Duplessis 310
Personalizing AI for Co-Creative Music Composition from Melody to Structure – Ken Déguernel, Mathieu Giraud, Richard Groult, and Sébastien Gulluni 317
Music-Robust Automatic Lyrics Transcription of Polyphonic Music – Xiaoxue Gao, Chitralekha Gupta, and Haizhou Li 325

SMC-22 Paper Session 5 **333**
Multisensory Integration Design in Music for Cochlear Implant Users – Razvan Paisa, Doga Cavdir, Francesco Ganis, Peter Williams, and Stefania Serafin 334
Generating Multiple Hierarchical Segmentations of Music Sequences Using Correlative Matrices – Paul Lascabettes, Corentin Guichaoua, and Elaine Chew 342
Deep HRTF Encoding & Interpolation - Exploring Spatial Correlations using Convolutional Neural Networks – Devansh Zurale, Shahrokh Yadegari, and Shlomo Dubnov 350
A Differentiable Neural Network Approach to Parameter Estimation of Reverberation – Søren V K Lyster and Cumhur Erkut 358
StyleWaveGAN: Style-Based Synthesis of Drum Sounds With Extensive Controls Using Generative Adversarial Networks – Antoine Lavault, Axel Roebel, and Matthieu Voiry 365
A Web Platform to Extract and Investigate Music Genre Labels in Spotify – Adriano Baratè and Ludovico Luca Andrea 373
Semi-Supervised Convolutive NMF for Automatic Music Transcription – Haoran Wu, Axel Marmoret, and Jérémy Cohen 381
Defending a Music Recommender Against Hubness-Based Adversarial Attacks – Katharina Prinz, Arthur Flexer, and Gerhard Widmer 389
Audio Metaphor 2.0: An Improved Classification and Segmentation Pipeline for Generative Sound Design Systems – Joshua Kranabetter, Craig Carpenter, Renaud Bougueng Tchemeube, Philippe Pasquier, and Miles Thorogood 395
Scream Detection in Heavy Metal Music – Vedant Kalbag and Alexander Lerch 403

SMC-22 Paper Session 6 **411**
Ciaramella: a Synchronous Data Flow Programming Language for Audio DSP – Paolo Marrone, Stefano D'Angelo, Federico Fontana, Gennaro Costagliola, and Gabriele Puppis 412
Count-Me-In: A Collaborative Step Sequencer for Audience Participation – Juan Pablo Carrascal 420

Designing the Balance Between Sound and Touch: Methods for Multimodal Composition – Claire Richards, Roland Cahen, and Nicolas Misdariis . . . 426
Analysis and Evaluation of Visual Cues in Graphical Interpolators – Darrell Gibson and Richard Polfreman 434
The Teinophon – Leo Fogadic and Daniel Overholt . . . 442
A New Sensor Technology for the Sonification of Proximity and Touch in Closed-Loop Auditory Interaction – Pascal Staudt, Anton Kogge, Marcello Lussana, Marta Rizzonelli, Benjamin Stahl, and Jin Hyun Kim 446
The Spatbox: An Intuitive Trajectory Engine To Spatialize Sound – Pierre Lecomte . . . 454
A Gaze-Driven Digital Interface for Musical Expression Based on Real-time Physical Modelling Synthesis – Devansh Kandpal, Prithvi Ravi Kantan, and Stefania Serafin . . . 461
The Singing Shower: A Melody-Sensitive Interface for Physical Interaction and Efficient Energy Consumption – Yann Seznec and Sandra Pauletto . . . 469
Design Concepts for Gaze-based Accessible Digital Musical Instruments – Nicola Davanzo and Federico Avanzini . . . 477

SMC-22 Paper Session 7 **484**
Design of Timbre With Cellular Automata and B-Spline Interpolation – Paul Lanthier and Matthew Klassen 485
Visualization of Deep Audio Embeddings for Music Exploration and Rediscovery – Philip Tovstogan, Xavier Serra, and Dmitry Bogdanov . . . 493
Polyfōnía: an Interactive Vocal Processing System using Finger Tracking – Natalia Kotsani and Areti Andreopoulou . . . 501
Zero-Cost Abstractions to Manage Audio Resource Allocation in Functional Programming Languages – Mike Solomon . . . 509
Making Computer Music on the Web with JSPatcher – Shihong Ren, Laurent Pottier, Michel Buffa, and Yang Yu . . . 516
A Poly-Temporal Programming Environment for Live Shows and Interactive Installations – Martin Fouilleul, Jean-Louis Giavitto, and Jean Bresson . . . 524
Measuring Virtual Audiences with The Musiclab App: Proof of Concept – Dana Swarbrick, Finn Upham, Cagri Erdem, Alexander Refsum Jensenius, and Jonna Vuoskoski . . . 532
Designing the (re)context – Jos Mulder . . . 540
TARTYP Generative Grammars and the Analysis of Hip Hop Beats – Israel Neuman . . . 546

SMC-22 Paper Session 8 **554**
Towards an FPGA-Based Compilation Flow for Ultra-Low Latency Audio Signal Processing – Maxime Popoff, Romain Michon, Tanguy Risset, Yann Orlarey, and Stéphane Letz . . . 555
Preserving, Restoring, and Passing Down Video-Game Music from the Past: The Case of DirectMusic – Luca Andrea Ludovico, Alberto Mattea, and Davide Andrea Mauro . . . 563
Designing Sound Representations for Musicology – Pierre Couprie . . . 570
Annotating Symbolic Texture in Piano Music: a Formal Syntax – Louis Couturier, Louis Bigo, and Florence Levé . . . 577
Exploring the Possibilities of Music Therapy in Virtual Reality: Social Skills Training for Adolescents with Autism Spectrum Disorder – Linnea Bjerregaard Pedersen, Ali Adjorlu, Valentin Bauer, and Stefania Serafin . . . 585
Barwise Compression Schemes For Audio-Based Music Structure Analysis – Axel Marmoret, Jérémy Cohen, and Frédéric Bimbot . . . 593
Formulating First Species Counterpoint With Integer Programming – Tsubasa Tanaka . . . 601
Guitar Improvisations with Hexaphonic Multieffect (Gihme) Dataset and Practice Analysis – Loïc Reboursiere, Thierry Dutoit, and Vincent Tiffon . . . 608

SMC-22 JIM Jeune Chercheur Papers **616**
Soutenir en classe l'écoute active, l'autonomie et l'échange en analyse musicale avec la plateforme web Dezrann – Alice Sauda, Mathieu Giraud, and Emmanuel Leguy . . . 617
Prototype de génération procédurale de contrepoints à la manière de Jean-Sébastien Bach – Jérémie Roux and Violaine Prince . . . 624
Espaces sonores paradoxaux: approche des harmoniques hypersphériques et implémentation dans une maquette logicielle, pour une pratique créative de la spatialisation immersive du son en free-party – Daniel Picciola . . . 634
Les instruments de musique numériques pour les personnes en situation de handicap cognitif et mental : exemples d'applications via le web – Matteo Olivo . . . 643
Estimation de paramètres de resynthèse de sons d'instruments de musique avec des outils de morphologie mathématique – Gonzalo Romero-García, Isabelle Bloch, and Carlos Agón . . . 653

SMC-22 Remote Demo Session **663**
Sound Scope Pad – Masatoshi Hamanaka . . . 664
Distracted - A piece for Violin and an Audio-Visual Interface – Anna Savery . . . 666

SMC-22 Demo Session 1 **668**
Spatialized Polyphonic Granular – Emmanouil Dimogerontakis . . . 669
Social Musical gaming with Count-Me-In – Juan Pablo Carrascal . . . 671
Playing the Virtual Glass Armonica – Astrid Pedersen, Morten Jørgensen, Halfdan Isaksen, and Mikkel Riedel . . . 672
AIM-Package : a Modular Standard for Structuring Patches in Max – Vincent Goudard . . . 674
The Development of Real-Time Movement Sonification System for Exercise Learning – Cherelle Connor . 675
A Virtual Reality Volumetric Music Video: Featuring New Pagans – Gareth W. Young, Neill O'Dwyer, and Aljosa Smolic . . . 677
MusCIcal Adventures: Virtual Reality as a Musical Training Program – Emil Sønderskov Hansen, Signe Marie Kromann Kristiansen, Ali Adjorlu, and Stefania Serafin . . . 679
Somax 2, a Reactive Multi-Agent Environment for Co-Improvisation – Joakim Borg, Mikhail Malt, and Gérard Assayg . . . 681
Pd on an IOT-Class Processor – Miller Puckette and Kerry Hagan . . . 683

SMC-22 Demo Session 2 **685**
Tickle Tuner - Haptic Smartphone Cover for Cochlear Implant Users' Musical Training – Francesco Ganis, Stefania Serafin, and Marianna Vatti . . . 686
Scramble Live: Combining LSTM and Markov Chains for Real-time Musical Interaction – Nicola Privato, Omar Rampado, and Alberto Novello . . . 688
Wavetable-Inspired Artificial Neural Network Synthesis – Nicholas Solem . . . 690
Creating Experiments With CosmoNote: Advancing Web-based Annotations For Performed Music – Daniel Bedoya, Lawrence Fyfe, and Elaine Chew . . . 692
Calliope: A Co-creative Interface for Multi-Track Music Generation – Renaud Bougueng Tchemeube, Jeff Ens, and Philippe Pasquier . . . 694
An Interactive Music Infilling Interface for Pop Music Composition – Rui Guo . . . 695
Reconstructing Prehistoric Music Technologies: Archaeological Explorations of Humans as Designers – Miriam Kolar . . . 697
Networked Performances With Ossia Score – Jean-Michaël Celerier and Pia Baltazar . . . 698
The Acceptable Range of Pitch Frequencies and Duration of Notes in Singing – Behnam Faghih and Joseph Timoney . . . 699
The Springamajig: A Flexible Digital Musical Instrument – Thomas Rushton and Dan Overholt . . . 700
Demos: Støjbox XYZ and Støjbox Spring – Benjamin Støier and Dan Overholt . . . 702
Kuplen: a Hands-On Physical Model – Gabriel Gustafsson, Marco Timossi, and Thomas Rushton . . . 704

IFC-22 Papers **706**
Feedback Acoustic Noise Control With Faust on FPGA: Application to Noise Reduction in Headphones – Loïc Alexandre, Pierre Lecomte, Marie-Annick Galland, and Maxime Popoff . . . 707
Automatically Generating Eurorack Hardware Running Faust Programs Using Eurorack-Blocks – Raphael Dingé and Stéphane Letz . . . 714
Envelope Following Via Cascaded Exponential Smoothers for Low-Distortion Limiting and Maximisation – Dario Sanfilippo . . . 725
Static Performance Analysis of Faust Programs Using Dataflow Modelling – Jaime Koh and Bruno Bodin 731
Faust and Category Theory – Nicholas Connell . . . 740
What's New in the Faust Ecosystem and Community? – Stéphane Letz, Romain Michon, and Yann Orlarey 750

Organizational Committee and Technical Team

Organizational Committee
Romain Michon (INRIA, France) – General Chair
Laurent Pottier (Université Jean Monnet, France) – General Chair
Yann Orlarey (GRAME-CNCM, France) – Scientific Chair
Constantin Basica (CCRMA, Stanford University, USA) – Music Chair
Clara Girousse (GRAME-CNCM, France) – Outreach Chair

Administrative Team
Myriam Chanudet (Université Jean Monnet) – Administrative Officer
Nadine Leveque-Lair (Université Jean Monnet) – Administrative Assistant

Technical Team
Pierre Cochard (INRIA) – Concerts Technical Director
Jules Dejardin (Université Jean Monnet) – Assistant Concerts Technical Director and RIM
Joseph Bizien (Université Jean Monnet) – Technical Assistant and RIM
Nascimo Marconnet (INRIA/Université Jean Monnet) – Technical Assistant
Maxime Popoff (INSA Lyon) – Video Streaming
Stéphane Letz (GRAME-CNCM) – IFC Coordinator
Philippe Landrivon (Université Jean Monnet) – AV Director
Pierre Lecomte (École Centrale de Lyon) – Ambisonics Specialist
Jean-Basile Sosa (Independent) – Ambisonics Specialist
Raphaël Forment (Université Jean Monnet) – Live Coding Concert Coordinator
Jade Jen (Université Jean Monnet) – Registration Front Desk Assistant

Volunteers
Matteo Olivo (Université Jean Monnet)
Luna Valentin (Université Jean Monnet)
Rémy Georges (Université Jean Monnet)
Thomas Rushton (Aalborg University)

Scientific Committee

General and Scientific Chairs
Romain Michon – INRIA, France
Laurent Pottier – Université Jean Monnet, Saint-Étienne, France
Yann Orlarey – GRAME, Centre National de Création Musical, France

Scientific Committee
Federico Avanzini – University of Milan, Italy
Alain Bonardi – Université Paris 8, France
Jean Bresson – Ableton, Germany
Mathieu Giraud – CNRS, CRIStAL, Université Lille, Inria Lille, France
Sylvain Marchand – University of La Rochelle, France
Georgios Marentakis – Østfold University College, Norway
Andrew McPherson – Queen Mary University of London, UK
Juhan Nam – Korea Advanced Institute of Science and Technology, South Korea
Sandra Pauletto – KTH Royal Institute of Technology, Sweden
Stefania Serafin – Aalborg University, Denmark
Sebastian Schlecht – International Audio Laboratories, Erlangen, Germany
Louena Shtrepi – Politecnico di Torino, Italy
Isabelle Viaud-Delmon – IRCAM-CNRS, France

Panel of Reviewers for the Main Scientific Track
Adrián Barahona-Ríos – University of York
Adriano Baratè – LIM
Jonathan Bell – Aix-Marseille University
Edgar Berdahl – Louisiana State University
Eoin Callery – CCRMA (Stanford University)
Elliot Canfield-Dafilou – Institut Jean le Rond d'Alembert, Sorbonne Université/CNRS
F. Amílcar Cardoso – DEI, CISUC, University of Coimbra
Alexander Carôt – HS Anhalt
Thibaut Carpentier – Ircam
Chris Chafe – CCRMA, Stanford University
Vasileios Chatziioannou – Institute of Music Acoustics
Stefano D'Angelo – dangelo.audio
Roger Dannenberg – Carnegie Mellon University
Matthew Davies – University of Coimbra
Florent de Dinechin – CITI, INSA de Lyon, université de Lyon, INRIA
Yuri De Pra – Centro E. Piaggio. University of Pisa
Stefano Delle Monache – Delft University of Technology
Michele Ducceschi – Univeristy of Edinburgh
Cagri Erdem RITMO – University of Oslo
Fabian Esqueda – Aalto University
Georg Essl – University of Wisconsin - Milwaukee
Leonardo Fierro – Aalto University
Federico Fontana – University of Udine
Francesco Foscarin – CNAM
Emma Frid – KTH Sound and Music Computing
Daniele Ghisi – IRCAM
Jean-Louis Giavitto – IRCAM - UMR STMS 9912 CNRS
Vincent Goudard – Collegium Musicæ
Andrew Horner – The Hong Kong University of Science and Technology
Daniel Hug – Zurich university of the arts
Hanna Järveläinen – Zurich University of the Arts
Alexander Refsum Jensenius – University of Oslo
Pierre Jouvelot – MINES ParisTech, PSL Research University
Haruhiro Katayose – Kwansei Gakuin University
Miriam Kolar – Amherst College
Olivier Lartillot – RITMO, University of Oslo
Stéphane Letz – Grame
Florence Leve – MIS - UPJV
Hans Lindetorp – Kungl. Musikhögskolan
Luca Andrea Ludovico – University of Milan

Raul Masu – ITI/LARSyS, FCT/NOVA
Benjamin Matuszewski – IRCAM
Davide Andrea Mauro – Marshall University
Doug McCausland – Doctoral fellow at Stanford University / CCRMA
Nicolas Misdariis – IRCAM
Fabio Morreale – The University of Auckland
Michael Mulshine – Stanford University
Hiroki Nishino – Chang Gung University, Taiwan
Stavros Ntalampiras – University of Milan
Daniel Overholt – Aalborg University
Elif Ozcan – Delft University of Technology / Faculty of Industrial Design Engineering
Razvan Paisa – Aalborg University
Rui Pedro Paiva – University of Coimbra
Claudio Panariello – KTH Royal Institute of Technology
Stefano Papetti – Zurich University of the Arts / ICST
Julian Parker – Native Instruments
Johan Pauwels – Imperial College London
Karolina Prawda – Aalto University
Giorgio Presti – University of Milan
Niccolò Pretto – Institute for Computational Linguistics, CNR, Pisa
Marcelo Queiroz – Computer Science Department – University of São Paulo
Mark Rau – McGill University
Tanguy Risset – Citi, INSA-Lyon
Davide Rocchesso – Università degli Studi di Palermo
Gerard Roma – Leeds Trinity Uiversity
Dirk Roosenburg – Oberlin College and Conservatory
David Roze – STMS lab, CNRS-IRCAM-SU
Mark Sandler – Queen Mary University of London
Giovanni Santini – privato
Diemo Schwarz – Ircam - CNRS STMS
Bernard Serpette – INRIA
Federico Simonetta – University of Milan
Julius Smith – Stanford University
Domenico Stefani – University Of Trento
Joao Svidzinski – Uni Paris 8
Burak Tamer – Bahcesehir University
Satoshi Tojo – JAIST
Luca Turchet – University of Trento, Department of Information Engineering and Computer Science
Andrea Valle – CIRMA - Università di Torino
Maarten Van Walstijn – Queen's University Belfast
Silvin Willemsen – Aalborg University
Minz Won Music Technology Group – Universitat Pompeu Fabra
Asterios Zacharakis – Aristotle University of Thessaloniki

Panel of Reviewers for the JIM/Young Researcher Track
Feron Francois-Xavier – IRCAM
Mathieu Giraud – CNRS, CRIStAL, Université Lille, Inria Lille
Vincent Goudard – Collegium Musicæ
Maxence Larrieu – Université Paris-Est
Florence Leve – MIS - UPJV
Eric Maestri – CICM Université Paris 8
Sylvain Marchand – University of La Rochelle
Bernard Serpette – INRIA

Panel of Reviewers for the International Faust Conference (IFC-22)
Alain Bonardi – Université Paris 8
Chris Chafe – CCRMA, Stanford University
Pierre Guillot – CICM - Université Paris 8
Pierre Jouvelot – MINES ParisTech, PSL Research University
Stéphane Letz – Grame
Julius Smith – CCRMA, Stanford University
Joao Svidzinski – Uni Paris 8

Artistic Committee

General and Artistic Chairs
Constantin Basica – CCRMA, Stanford University, USA
Romain Michon – INRIA, France
Laurent Pottier – Université Jean Monnet, Saint-Étienne, France

Artistic Committee of the Call for Mixed Music for Large Ensemble
Constantin Basica – CCRMA, Stanford University
Romain Michon – INRIA
Laurent Pottier – Université Jean Monnet
Bruno Montovani – Ensemble Orchestral Contemporain
Sebastian Rivas – GRAME, Centre National de Création Musicale

Artistic Committe of the Call for Chamber Mixed Music
Pascale Jakubowski – Conservatoire de Saint-Étienne
Luce Zurita – Conservatoire de Saint-Étienne
Sébastien Giebler – Conservatoire de Saint-Étienne
Hervé Cligniez – Conservatoire de Saint-Étienne
Elisabeth Gaudard – Conservatoire de Saint-Étienne
Marianne Gaiffe – Conservatoire de Saint-Étienne
Pascal Jemain – Conservatoire de Saint-Étienne
Jérôme Bertrand – Conservatoire de Saint-Étienne
Lyuba Zhecheva – Conservatoire de Saint-Étienne
Léonard Chantepy – Conservatoire de Saint-Étienne
Nicolas Domingues – Conservatoire de Saint-Étienne
Hamid Medjbeur – Conservatoire de Saint-Étienne
Gilles Peseyre – Conservatoire de Saint-Étienne
Didier Muhleisen – Conservatoire de Saint-Étienne
Safia Azzoug – Conservatoire de Saint-Étienne
Claire Lapalu – Conservatoire de Saint-Étienne

Panel of Reviewers of the Main Artistic Track
Freida Abtan
Sabina Hyoju Ahn – Kunstuniversität Linz
Stefano Bassanese – Conservatorio di Torino
Jonathan Bell – Aix-Marseille University
Christopher Biggs – Western Michigan University
Bálint Bolcsó – University of Pécs / Liszt Academy of Music, Budapest
Myriam Boucher – Université de Montréal
Michael Boyd – Chatham University
Eoin Callery – CCRMA (Stanford University)
Carole Chargueron – Escuela Superior de Música
Alex Chechile – Stanford University
Natacha Diels – University of Pennsylvania
Kaley Eaton – Cornish College of the Arts
Kerry Hagan – University of Limerick
Anne Hege – Mills College
Cat Hope – Monash University
Steven Kemper – Mason Gross School of the Arts, Rutgers University
Shelly Knotts – Network Music Festival/Durham University
Juraj Kojs – University of Miami
Panayiotis Kokoras – University of North Texas
Alexandros Kontogeorgakopoulos – Cardiff School of Art and Design, Cardiff Metropolitan University
Silvia Lanzalone – CRM-Centro Ricerche Musicali Roma
Nolan Lem – Stanford
Andres Lewin-Richter – Phonos Foundation
Jia Li – Shcmusic
Michelle Lou – University of California, Santa Cruz
Thessia Machado – independent artist
Eric Maestri – CICM Université Paris 8
Douglas McCausland – Doctoral fellow at Stanford University / CCRMA
Caroline Miller – Portland State University

Dafna Naphtali
Adolfo Núñez – ECIS-CSIPM
Joo Won Park – Wayne State University
Sam Pluta – University of Chicago
Leah Reid – University of Virginia
Margaret Schedel – Stony Brook University
Franziska Schroeder – Queen's University Belfast
Scott Smallwood – University of Alberta
Akira Takaoka – J. F. Oberlin University
Hans Tammen – School of Visual Arts
Giuseppe Torre – University of Limerick
Pierre Alexandre Tremblay – University of Huddersfield
Davor Branimir Vincze – Novalis Concept
Sarah Weaver
Rafal Zapala – Academy of Music in Poznań

Artistic Program

Concert at *The Panassa at the Comète* – June 7, 2022, 8pm

- *Somax 2, a Reactive Multi-Agent Environment for Co-Improvisation* – Mikhail Malt, Benny Sluchin, and Simone Conforti
- *Sounding Microcosmos* – Serkan Sevilgen and İpek Oskay
- *Elegy (Ready, Set, Rapture)* – Rob Hamilton
- *The Core* – Klaus Scheuermann, Constantin Basica, Chris Chafe, Fernando Lopez-Lezcano, Henrik von Coler and Juan Parra
- *The Luminosphere project - Brass distortion* – Joseph Bizien
- *Fight with Shattered Instruments in the Valley of the Shattered Idols* – Gordon Delap
- *Void ii* – Kari Vakeva
- *Primo Vere* – Alessandro De Cecco

Concert at *The Corbusier Church* of Firminy – June 8, 2022, 7:30pm

- *Hadrosaur Variations II* – Courtney Brown
- *La naissance de la lumière à partir de l'esprit du feu* – Christian Dimpker
- *Sonic Luster* – Kimia Koochakzadeh-Yazdi
- *Intuitive Access* – Se-Lien Chuang and Andreas Weixler
- *I am an orchid* – Juraj Kojs

Concert at *The Théâtre Copeau* in Partnership With Saint-Étienne Conservatoire – June 9, 2022, 6:30pm

- *Les possibilités des friches* – Diane Schuh
- *Pulsar [Variant I]* – Seth Shafer
- *Not Welcome* – Marie-Hélène Bernard
- *hlör u fang axaxaxas mlö* – Oded Ben-Tal
- *Replication* – Michael Boyd

Concert at *The Théâtre Copeau* in Partnership With the Ensemble Orchestral Contemporain – June 9, 2022, 8:00pm

- *Au seuil des cimes innombrables* – Victor Malcey, Rafaël Carosi and Jules Dejardin
- *Quimera de un recuerdo* – Luis Alpiste et Joseph Bizien
- *Nei rami chiari* – Lara Morciano
- *Diffrakt* – Adrien Trybucki

Concert at *The Centre des savoirs pour l'innovation* (Remote Fixed Media Pieces) – June 10, 2022, 9:00am

- *Ethereal Transference (ver. 2)* – Ji Won Yoon and Woon Seung Yeo
- *Pulse Geometries* – Marcel Zaes
- *Around Space* – Philippe-Aubert Gauthier and Tanya St-Pierre
- *La Sombra* – Sebastian Pafundo
- *Alternate spaces* – Gabriel Zales Ballivian
- *Strangely Familiar Faces (ver. 3)* – Ji Won Yoon and Woon Seung Yeo
- *The Eighth Island* – Lidia Zielinska
- *Stagnant spring, drifting inside* – Ayako Sato
- *Anamorphosis* – Clemens von Reusner

- *Eternal Accelerando* – Ryo Ikeshiro

Concert at *The Centre des savoirs pour l'innovation* (Remote Fixed Media Pieces) – June 10, 2022, 1:30pm

- *Estuaries 4* – Bret Battey
- *Poplite* – Francesco Bossi
- *Ice Door* – Juan Escudero
- *BLOCK* – Noah Berrie
- *Xeno* – Enrico Dorigatti
- *Another One* – Michael Boyd
- *Claret* – Rosalía Soria Luz
- *Trichromacy* – Andrew Watts
- *Rabbit Locked In the Kitchen* – Ayako Sato
- *No Future* – Giovanni Onorato
- *Paradigm Shift: Tapping into the Quantum Field* – Cecilia Suhr
- *Reverie* – Leah Reid
- *What Sleeps Beneath* – Kramer Elwell
- *Trippin' On the Edge of Time* – Alessio Mastrorillo
- *Time Garden: Skull Bridge* – Charles Nichols and Scotty Hardwig and Zach Duer
- *Construction in Kneading* – Ryo Ikeshiro
- *Psi* – Rodney Duplessis
- *DigiTral* – Georgios Varoustos and Matheos Zaharopoulos

Concert at *The FIL* (Remote Fixed Media Pieces) – June 10, 2022, 8:00pm

- *Departure* – Dong Zhou
- *Contested Equilibrium* – Klaus Scheuermann and Dani Gal
- *People are the Knife; I am the Meat* – Dong Zhou and James Cheung
- *Still Life* – Robert Seaback
- *Aeon* – Miles Thorogood and Dulic Aleksandra
- *Extrapolation de la campagne* – Jean-Basile Sosa
- *Special Live Coding Session*

Installations

- *A Walk to Meryton* – Arne Eigenfeldt

SMC-22 Preface

By Romain Michon and Laurent Pottier, SMC-22 General Chairs.

Welcome to the 19th Sound and Music Computing conference (SMC-22)!

We have the privilege to organize the first in-person edition of SMC since the beginning of the COVID-19 pandemic two years ago. Planning within this context produces many uncertainties and makes our task very challenging. Thanks to our fantastic team and the support of a wide range of partners, we have devised a fun and interesting event where attendees may celebrate both the enduring music research of our field as well as the end of two years of social distancing.

SMC-22 is taking place in Saint-Étienne on June 5-12th, 2022 and is organized under the auspices of the ECLLA Lab of the Université Jean Monnet of Saint-Étienne, the Emeraude Team at INRIA (National Institute for Research in Digital Science and Technology), GRAME - Centre National de Création Musicale, and the SMC steering committee. It combines three conferences in one: the International Faust Conference (IFC-22), the Journées de l'Informatique Musicale (JIM-22), and SMC-22. These three conferences are unified under one common theme: Music Technology and Design. SMC-22 is happening in a wide range of venues throughout the city of Saint-Étienne: Télécom Saint-Étienne, the Centre des Savoirs pour l'Innovation, the Comète, Saint-Étienne Opera, the Corbusier Church, the FIL, etc., in partnership with a large number public and private institutions.

Saint-Étienne is an easily walkable city featuring fantastic facilities and venues. As one of France's main historical industrial hubs, design plays a central role in the identity of Saint-Étienne, hence the theme of this year's conference. In this context, SMC-22 has been integrated to the program of the 2022 Design Biennale taking place in Saint-Étienne between April 6th and July 31st, 2022.

SMC-22 received a total of 122 scientific papers, 6 young researcher papers, 23 demos, 19 lightning/industry talks, and 85 artistic works submissions out of which 80 scientific papers, 5 young researcher papers, 23 demos, 6 lightning/industry talks, and 53 artistic works were selected. IFC-22 received 6 paper submissions, all of which were all accepted.

With this curated and peer reviewed content we are hosting 3 days of summer school, 5 days for the scientific conference, and 6 concerts. We aim to make SMC-22 as accessible as possible by making the summer school as well as registration for people just auditing the conference completely free.

After two years of social distancing, we believe that the members of our community will benefit from a return to in-person interactions, and so we are promoting a new format for paper presentations that encourage increased discussion between the conferees. In this new format, authors present five-minute oral presentations and then provide more details in a subsequent poster session. We also have planned numerous social events to facilitate the collaborative spirit of our field.

The université Jean Monnet of Saint-Étienne offers a "Professional Computer Music Producer master's program." In this context, we are promoting mixed music pieces in the artistic call of the conference, offering collaborations with local acoustical instrument students and their teachers at Saint-Étienne Conservatory as well as with the internationally renowned EOC (Ensemble Orchestral Contemporain) contemporary music ensemble.

We are opening SMC-22 as much as possible to the public by making most of our concerts freely accessible and by organizing "Faites du son," a three-day parallel event featuring short talks and workshops targeting a broad audience as well as local schools.

SMC-22 can be attended both in-person and remotely. While this hybrid format makes the conference more accessible, it also produces many uncertainties with our budget. Available funds were only known a few weeks before the conference, when we finally learned the number of people attending the events in-person. With equally late notice we learned that 70% of the selected artistic works that have been accepted at the conference will be performed remotely (while 90% of scientific submission will be presented in person). As we do not want to impose too many remote fixed-media concerts on our in-person attendees, we must completely reorganize the concert schedule at the last minute. While technology now allows us to watch a concert from the comfort of our living room – which is fantastic – in-person concerts featuring pieces performed remotely can be tedious if the music was not devised to be experienced in this modality. This might be worth noting by future organizers of SMC who must choose whether to offer the hybrid format.

The COVID-19 pandemic has significantly accelerated the transition to a world where social interactions happen virtually. Driven in part by critical health initiatives, this change may also alter the way institutions allocate funds for conference travel, a trend that may extend beyond the pandemic; indeed, we heard numerous stories of people with accepted submissions unable to make it to the conference in-person because their institutions asked

them to only go to remote conferences for cost-efficiency. We need to consider the implications of this trend, as it is difficult to replace the networking facilitated by in-person interactions. The organizers of SMC 2022 believe that the synergies that arise from conversing in a poster session, discussing a concert at dinner, or devising collaborative research over a glass of wine, play a decisive role in the vital future of sound and music computing.

Long live in-person conferences! Long live SMC!

IFC-22 Preface

By Yann Orlarey, Scientific Director of GRAME-CNCM and Father of Faust.

Faust is now 20 years old. The idea was born in July 2001, during the Libre Software Meeting (LSM) in Bordeaux. With Myriam Desainte-Catherine, we organized the "Computer Music" session of this meeting. On this occasion, we proposed to François Déchelle and Norbert Schnell, researchers at IRCAM, to present the jMAX project, a free version (written in JAVA language) of the MAX language.

The Starting Point

During our discussions, we talked, among other things, about MAX patches, in which messages and audio signals flow from top to bottom. The idea for Faust came from a reversal of perspective that is quite difficult to explain in words. But rotating a Max patch 180 degrees makes it look like a syntax tree. This encourages a much more declarative view of things, where the *entire thing*, at the top, breaks down into parts, down to the leaves that are signals. A dynamic computation on samples becomes a static computation on infinite signals that can be composed thanks to algebraic operators.

Since this initial intuition, a lot of ground has been covered, thanks to the hard work of the GRAME research team, but also thanks to fantastic contributors and a very dynamic, creative, and committed user community.

Adding Meta-Programming Capabilities

A significant breakthrough in the expressiveness of the language was achieved when, with Albert Gräf, we designed and implemented the pattern matching mechanism borrowed from his Q language. With this technique, complex signal processing can be described algorithmically in a few lines of Faust code. Julius Smith's implementation of the FFT algorithm in Faust is a remarkable example of that.

Today, Faust allows musicians to develop their own plugins, and even complete audio applications, on multiple platforms, without the complexity of traditional C or C++ development, and at a fraction of the time and cost of such development. Special attention has always been paid to the expressiveness of the language. The Faust language should not only help us code, but it should also help us think, design, and invent.

As a result, even very experienced computer scientists with a perfect mastery of C or C++ can benefit from the productivity gains afforded by Faust, without sacrificing performance thanks to the quality of the code produced by the compiler, which rivals handwritten C/C++ code.

Faust Libraries

The Faust libraries are becoming increasingly complete, thanks to the absolutely decisive work undertaken for many years by Julius Smith. Romain Michon has completely re-architected and documented them, which has since allowed for many external contributions. These libraries contribute greatly to Faust's productivity and attractiveness.

An interesting effect of the meta-programming capabilities is the development of new Domain Specific Languages inside the Faust DSL itself. This is the case with the physmodels.lib (Faust physical modeling library) or the wdmodels.lib which allows us to build Wave Digital Filters. These libraries are defining new algebras to build complex models starting from basic building blocks and specific connection rules.

Architectures Files

Another element contributing to this attractiveness is the numerous architectures on which Faust code can be deployed, from VST plug-ins on Windows, macOS, or Linux, to ESP32 microcontrollers, audio applications for the Web or smartphones, and established audio frameworks like JUCE.

More than a hundred deployment options are available today. Thanks to the FAST project led by Romain Michon, and the Emeraude team directed by Tanguy Risset, Faust can even be used to program FPGAs for ultra-low latency audio applications.

Today, a company wishing to launch a programmable audio card will likely gain a lot to offer a Faust architecture for its hardware. It is a minimal investment which in return allows the manufacturer to benefit from the entire catalog of Faust programs and libraries for his card. Of course, the Faust community also gains from having a new architecture at its disposal.

This snowball effect works all the better as Faust tools are easily accessible, especially from the web with the excellent IDE by Shihong Ren, but also from most music programming environments (i.e., Max, Csound, Puredata, etc.). This was made possible by the remarkable work of reorganizing the Faust compiler led by Stéphane Letz. In addition to the C++ backend, the compiler now has backends for many programming languages, including WebAssembly and LLVM-IR.

Library Version of the Compiler

In addition, the compiler also exists as a library version. Thanks to this, and the use of backends like WebAssembly and LLVM-IR, it is possible to embed Faust in many applications, do dynamic compilation from DSP programs to executable code, in a web browser or in the aforementioned music programming environments. To make Faust easy to compile on all platforms, the build process was at one point completely redesigned by Dominique Fober using CMake.

Part of this work of diversification of Faust tools was carried out within the framework of the ANR Feever project, coordinated by Pierre Jouvelot, and carried out in collaboration with the Ecole des Mines de Paris and the Université Jean Monnet of Saint Étienne.

Learning Faust

This has had a very positive impact on the teaching of Faust. Today, one of the most important places for this teaching is certainly the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University, with courses given by Julius Smith and Romain Michon. Thanks to this, some DSP engineers in Silicon Valley are now trained in Faust! In France, we can also mention Paris 8 with Alain Bonardi's courses, the Master ATIAM at IRCAM, or the Master of Computer Music directed by Laurent Pottier in Saint-Étienne.

Artistic and Industrial Applications

Faust has been used in dozens of artistic projects. In particular, Christophe Lebreton has developed the SmartFaust project, a concept of musical applications for smartphones, only using the user's gestures to control the instruments. Other musical projects are using the faustgen~ Max/MSP external, allowing for the embedding of the Faust compiler in the patch and creating tailor-made DSP blocks. The Faust Web version has been integrated in the GRAME INScore project, and a lot of other use-cases have been experimented.

Several companies are now using Faust to develop the DSP part of their audio applications, released in commercial products. We can cite in particular GeoShred by Moforte, Noisy of Expressive E, Drumbox by Little Robots, Motion Vox, etc., and many others at the prototyping stage.

Expanding the Community

Animating the growing community requires new tools to better follow and help the users. A dedicated Faust Slack channel has been opened, as well as a `#faust` channel in The Audio Programmer community.

More than 110 projects using Faust in different ways — I.e., musical pieces, artistic projects, plugins, standalone applications, integration in audio programming environments, development tools, research projects, embedded devices, Web applications, etc. — are now listed on this page:
`https://faust.grame.fr/community/made-with-faust/`

Future Research and Developments

In the years to come, all these approaches will, of course, be pursued. But other areas are opening up. We have mentioned compilation for FPGA. To be fully effective, it requires the ability to express fixed-point computations. This in turn implies being able to determine the optimal fixed point format for each signal and each internal operation.

Another area of work is to be able to control precisely when and at what speed computations are carried out. One of the approaches pursued in the past was to explicitly introduce upsampling and downsampling operations on the signals. The current approach, with on-demand, considers that these operations should not be performed on the signals, but on the signal processors themselves, which simplifies a lot of things.

In the mid-term, another area of work being considered is *data parallelism*. Faust is a naturally parallel language. The user does not have to do anything to declare parallelism, unlike a sequential language such as C. On the other hand, data parallelism, which consists in detecting that the same computation is done in parallel on different

data, which could therefore be grouped in vectors to benefit from SIMD instructions, is not currently exploited. There will probably be significant gains in terms of efficiency, especially for computations performed on a single core.

So, happy 20th birthday Faust! And a very happy IFC to all.

Keynotes

Ge Wang – Artful Design: Technology in Search of the Sublime

How might the shaping of technology make our lives more expressive and meaningful? What does it mean to design well — to design ethically? This presentation is a meditation on the design of everyday tools, toys, musical instruments, games, and social experiences. It is a story of how we shape technology – and how technology, in turn, shapes us and our future.

About the Speaker
Ge Wang is an Associate Professor at Stanford University's Center for Computer Research in Music and Acoustics (CCRMA). He researches the artful design of tools, toys, games, instruments, programming languages, virtual reality experiences, and interactive AI systems with humans in the loop. Ge is the architect of the ChucK audio programming language and the director of the Stanford Laptop Orchestra. He is the Co-founder of Smule and the designer of the Ocarina and Magic Piano apps for mobile phones. A 2016 Guggenheim Fellow, Ge is the author of /Artful Design: Technology in Search of the Sublime/, a photo comic book about how we shape technology – and how technology shapes us.
`https://ccrma.stanford.edu/~ge/`

Yann Orlarey – From Combinatory Logic and Lambda

If the first programming languages appeared in the 1950s, precursor formalisms existed long before the very first computers. We can mention in particular the Lambda Calculus of Alonzo Church in the '30s, or, even earlier, in the '20s, the Combinatory Logic of Moses Schonfinkel.

One hundred years later, these proto-programming languages from before the very first computers are an extremely interesting source of inspiration for programming language designers. Some concepts, such as lambda (anonymous function) from Lambda Calculus, which were ignored for decades by the mainstream languages (with the exception of LISP), are now adopted by almost all modern programming languages.

During this keynote, I will show how these formalisms directly inspired the design of FAUST, in particular how combinatory logic is at the root of its syntax, and how the notion of abstraction of Lambda Calculus plays a crucial role in its programming model.

About the Speaker
Born in 1959 in France, Yann Orlarey is a composer, researcher, member of the Emeraude research team (INRIA, INSA, GRAME), and currently scientific director of GRAME, the national center for musical creation based in Lyon, France. His musical repertoire includes instrumental, mixed, and interactive works as well as sound installations. His research work focuses in particular on programming languages for music and sound creation. He is the author or co-author of several musical software, including the programming language FAUST, specialized in acoustic signal synthesis and processing.

Sasha Leitman – Nuance and Resonance: Physical Interaction Design and the Acoustic World

This presentation will explore the design of computer music controllers and musical interactions inspired by auditory resonance and the nuanced control of acoustic instruments. I will examine the converging trends that are finding inspiration in the nitty-gritty real world of resonance, embodiment, feedback, materiality and tactile response. From feedback musicianship to digital lutherie to haptic innovations, music makers are using reality to augment their computer music practice.

About the Speaker
Sasha Leitman is a sound artist, composer, inventor and educator from California. She has been making musical instruments, new interfaces for musical expression and sound art installations for the last 20 years. Her work often features elements of field recordings, found objects, DIY craftsmanship, underwater sound, physical play and a deep connection to the materials that make up our physical world. After spending over a decade teaching courses and managing the Max Lab – an Interface Prototyping Lab, at Stanford University's Center for Computer Research in Music and Acoustics (CCRMA), she decided it was time to try out the other side of the Pacific. She is currently living in the Wellington and Wairarapa regions of New Zealand, exploring the design of computer music controllers inspired by auditory resonance and the nuanced control of acoustic instruments. More information can be found on her website, `http://sashaleitman.com`.

Michel Buffa – Audio on the Web

In this keynote, Michel Buffa will tell you, and will show you -with the help of many live demonstrations-, the history of audio support on the Web platform. From Flash to HTML5, not forgetting aborted initiatives such as Mozilla's Audio Data API, you will see how the development of the W3C standards WebAudio, WebMidi, Web Assembly, Web Components (and a few others that are emerging at the moment, WebML, WebCodecs, etc.) has been able to transform the Web browser into a powerful tool for computer music. You'll also see that, as in the world of native applications, there are commercial digital audio workstations (DAWs), lots of open source applications, libraries, DSL programming languages that can compile and run in the browser, and that open source community standards for interoperable plugins exist and are trying to be adopted. You'll also see how the web platform stands out, often bringing a collaborative and instantaneous aspect that is more unusual in native applications. Indeed, we could not have predicted the huge shift to the online, web-based world that is now part of our daily lives after the COVID pandemic, but the expertise accumulated by WebAudio developers and academic researchers over the past decade makes the web today a perfectly placed platform to help shape how the music world continues to adapt to this new way of living and working.

About the Speaker
Michel Buffa is a professor/researcher at University Côte d'Azur, a member of the WIMMICS research group, common to INRIA and to the I3S Laboratory (CNRS). He contributed to the development of the WebAudio research field, since he participated in all WebAudio Conferences, being part of each program committee between 2015 and 2019. He actively works with the W3C WebAudio working group. With other researchers and developers he co-created a WebAudio Plugin standard. He has been the national coordinator of the french research project WASABI, that consists in building a 2M songs knowledge database that mixes metadata from Cultural, lyrics and audio analysis.
`http://users.polytech.unice.fr/~buffa/`

Presentation of Lightning and Industry Talks

Music(?) Artificial "Intelligence"

Oded Ben-Tal

The aim of this short talk is to interrogate recent advances in Music AI and ask how musical or intelligent those really are. Applications of Machine Learning to musical tasks including composing, performing, and producing, have produced surprisingly good results in many ways. At the same time the headlines around those often overhype these achievements and end up making claims about music which are disconnected from the knowledge and knowhow developed by musicians and music scholars. It is understandable that at the early stages of the development of this technology it is driven primarily by the capabilities and needs of the technology itself. But as the field matures, it is imperative that musical considerations rather then technological capabilities set the research agenda.

The starting point for this talk is my own involvement with utilising music machine learning for compositional purposes. The Folk-RNN project applies deep learning to a data set of Irish folk tunes. On the one hand the resulting model generates plausible outputs for the style. But as our work has shown, the learning in the machine is fragile, limited and breaks down easily in distinctly non-musical and unintelligent ways. This does not mean that the research is flawed or useless – the technology can be used creatively. But for the research to advance beyond the fun novelty effect requires including expert musicians into the research loop and asking more musical questions. For instance, we often prioritise quality over quantity in our evaluation of creative activities. Is this still true when a system can produce 100,000 or more tunes? And how do you evaluate the quality of 100,000 tunes at all?

One of the outcomes of engaging with machine learning and data-driven technology is the establishment the Datasounds, datasets and datasense research network which aims to identify core questions that will drive forward the next phase in data-rich music research, focused on creative music making. More data and new ways of utilising it open opportunities to study music and make music in novel ways. These challenge some of our inherited assumptions about creativity and creative processes, about talent and musicality, about expertise and music pedagogy. At the same time, the creativity of musicians could be used to interrogate the capabilities and limitations of these new technologies. This is particularly important with the 'black-box' models of much machine learning research. The network will provide a forum for multi-partite discussion among researchers from different fields and practitioners including performers, improvisers, composers, and other artists. Like the research network, the talk will focus on questions rather than aim to provide answers. Enabling a more integrated approach to music AI research that melds the knowledge and concerns of engineers, data scientists, musicians, and music scholars requires some rethinking on all sides and venturing out of established mode of work towards a shared understanding of the important research questions.

Introducing the Sound Corpus Survey: What's that I hear? Understanding Human Sound Description to Make Better Generative Audio Engines

Craig Carpenter, Joshua Kranabetter, Renaud Bougueng Tchemeube, Miles Thorogood, and Philippe Pasquier

In film, television, VR and games, sound often leads the image. Sound designers are experts at where to place the menacing crow or distant cry of a baby. In horror films when we hear the menacing drone, we prepare for the dramatic event it signals. Sound, whether we are aware or not, is frequently the emotional cue that leads viewers through a story. When searching for the perfect sound, designers interact with large audio databases that retrieve files based on characteristics and emotions often embedded in metadata. A big part of any sound designer's job is knowing their libraries inside and out. Expanding audio web databases demand designers to be more creative with their search queries. As a means of streamlining the time-consuming process of sound retrieval and composition, our team explores the emergent field of non-human sound event detection models. We want to learn more about how sound designers describe the experience of sounds when designing audio for media. Our ongoing sound corpus study will help us understand more about the kind of language used in the creative process and enable us to advance research in sound design and develop new production tools. The online study involves listening to 20 audio recordings then entering a tag for the feeling that the sound communicates and a tag to describe the character of the sound.

This talk presents and discusses preliminary results from a sound corpus study aimed to generate a lexicon of descriptors to aid AI and machine-learning in soundscape generation. The first comprehensive study of expert to beginning sound designers focussed on improving generative soundscape models, we employed Google's AudioSet taxonomy as it includes a wide spectrum of audio events. From human and animal sounds to natural and environmental sounds, to musical and miscellaneous sounds. As AudioSet's taxonomy is developed to cover the widest range of acoustic detection by "typical listeners" the qualitative data generated by the survey should yield a lexicon of human sound descriptors that serve to improve non-human acoustic event detection. From the audio descriptors and associated meanings, we will develop a lexicon that will address this gap in the literature. The

talk will present participants with samples from the survey, offering an opportunity to respond to the selections from survey's sound events. We will then present preliminary findings and discuss how they might be used in ongoing machine-learning and AI models for soundscape generation, briefly introducing participants to Audio Metaphor and other current models that bypass tagged audio files through the use of user-directed models of emotion prediction.

Kinetic Displays of Performers' Movement for Telematic Music Performance

John Granzow, Matt Albert, and Michael Gurevich

Telematic music encompasses a vibrant field of research and practice exploring the aesthetic, technical, and cultural implications of real-time music-making by people in disparate geographical locations. As various forms of remote musical collaboration have proliferated in the context of the COVID-19 pandemic, the need for physical isolation has brought into relief the degree to which music-making is reliant on sharing physical space. Colocated musicians employ a variety of explicit and implicit gestural cues to coordinate their actions—shared meanings can emerge from a subtle nod of head, an exaggerated rise of the torso, or a fleeting glance. Our previous experiments, and those of others, have shown that this ability for visual coordination among performers, and visualization of remote performers for audiences, are among the most pressing needs in telematic performance.
For some time, technologies such as jacktrip and advanced fiberoptic switching have enabled uncompressed CD-quality audio with sufficiently low latency to support synchronous rhythmic music performance between locations on the same continent. However, the encoding, transmission, and decoding of video signals can incur substantial latencies. Furthermore, there is evidence that video might not actually be the most effective way to support what musicians and audiences need. Research by Wendy Ju and others has shown that even at the same scale and information rate, physical movement in space is more engaging than video, and supports more effective interpretation of gestures. Video has limited ability to communicate complex human qualities such as "effort" and "tension" that are essential in music, but which involve extremely subtle and highly nuanced gestures or isometric muscle activations. Finally, video suffers from aesthetic deficiencies. In telematic performance, where we rely on the network to support metaphors of extending or sharing space, video only highlights spatial disjunction. We will always see artifacts of the distant environment; we'll never make it look or feel like we're sharing space, even if we can make it sound like it.
In this talk we report on the Visualizing Telematic Music Performance project, whose goal is to create digitally fabricated mechatronic displays—moving avatars or kinetic objects that display the actions and efforts of remote musicians in 3-dimensional, physical space. Following a practice-led approach, a multidisciplinary team of faculty and student researchers have employed marker-based infrared motion capture to encode, analyze, and transmit performers' movements. We describe a series of experimental screen-based visualizations of simple motion descriptors and other biomechanical sensing data, which in turn informed the development of novel mechatronic displays that have been employed in simulated telematic performances with live audiences. The development process, involving performers, music technologists, and engineers, has shed light on processes by which musicians adaptively develop shared gestural repertoires with human performers and their mechatronic counterparts. We demonstrate the project's outcomes thus far, and discuss the qualitative experiences of performers and audiences.

Enhancing Sound, Music, and Movement Computing with Differentiable DSP

Cumhur Erkut

Artificial Intelligence and Machine Learning play an increasingly significant role in Sound and Music Computing (SMC). Recently, the Differentiable Digital Signal Processing (DDSP) approach gave a first indication of how some ideas of differentiable programming can be applied to SMC. In my research on sonic and embodied interaction, I try to follow and operationalize the developments after DDSP, with a special focus on how data-flow programming languages for extended realities in sound, music, and movement computing can be designed, implemented, and deployed with differentiable programming. When there is enough data, computational power, good objectives, and metrics for specific tasks, Deep Learning structures can learn quite accurately without explicit programming. However, in the real-world of SMC, these ideal components rarely come together. Moreover, as the resources grow, specialize, and become varied day by day, it becomes a challenge to keep up, develop, evaluate, deploy, and maintain useful applications. Recently we operationalized differentiable frameworks for sound (Alonso and Erkut 2021; Ganis et al. 2021) and movement synthesis (Kaspersen et al. 2019). In this SMC Lightning talk, I will demonstrate three applications of sound (Alonso and Erkut 2021; Ganis et al. 2021) and movement synthesis (Kaspersen et al. 2019), and illustrate how emerging DDSP control-synthesis approaches relate to previous research from the last decade, yet are boosted by powerful automatic differentiation frameworks such as PyTorch and Tensorflow.

master_me – a Sound-Engineer Taps Into Open Source Coding

Klaus Scheuermann

20 years of working in the audio industry made me feel like a puppet on the strings of soft- and hardware giants. Apple, Pro tools, Steinberg – you name it. I wanted to earn back my sense of agency, and the first step was to understand how software is made. So I started learning Python – with few successes and little impact because, as I soon realized, I did not have a project that could carry me from general motivation to sense of purpose.
As the pandemic locked down life around the world, I joined a group of musicians, developers, and scientists who started the CCRMA-hosted "Quarantine Sessions," a distributed network electro-acoustic performance created via jacktrip, jitsi and obs. With the first lockdown in Germany, a new window to the world opened up for me.
As is often the case in analogue live shows, soundchecks for the "Quarantine Sessions" were always great, but our passion and enthusiasm during the shows pushed levels into uncomfortable regions. As a mastering engineer, I knew what had to be done. I just did not know how to do it. The truly transdiciplinary expertise in our band pointed me to the faust programming language, and an inspiring journey began.
Since that Summer in 2020, my automatic mastering software master_me has taken care of levels, dynamics and distortion in almost 50 "Quarantine Sessions" and has been successfully used for other streaming events as well. With the help of the amazing Faust community, it evolved into a usable, stable tool for every online event's master bus.
But if it works for music, why not for speech? During the pandemic, a lot of great content conferences were forced into the digital realm – and sounded pretty bad. It's pretty hard to concentrate when you constantly have to adjust your headphone and speaker levels manually.
So I developed the concept of a software dedicated to speech sound optimization and received state funding for open-source development by the German Ministry of Education and Research. Beginning this March, I will be developing sounds_good – a software solution that will work more smoothly than its clumsy working title suggests.
My lightning talk aims to be a motivation to people who are new to coding, struggle with doubts, or have no idea where to start. It briefly sketches the steps of my own learning process and portrays my journey from "just sound engineer" to "sound-engineer with a coding hobby" – one that helps the open-source world sound a little better.

Latency, Interaction Bandwidth, and Why They Matter

Andrew McPherson, Robert Jack, Giulio Moro, S. M. Astrid Bin, and Adan Benito

Real-time audio synthesis grows ever more sophisticated in both academia and industry, driven by advances in computing power, but these advances have not been accompanied by a similar improvement in richness of interaction. It is common for even the most sophisticated numerical models of acoustic instruments or analog circuits to be restricted to a small number of audio inputs and outputs, or they might be controlled symbolically through MIDI or similar standards. It is thus possible to create elaborate digital simulations of both familiar and novel instruments, but the performer's embodied engagement with them lags far behind.
The dominant constraint hindering digital instruments is no longer computational power, but the bottleneck of physical-digital integration. In contrast to acoustic instruments where the entire object can contribute (however subtly) to interaction and sound production, digital instruments often rely on just a few sensors within an otherwise passive physical structure. Interaction design is then framed as a data mapping problem, but many of the important decisions have been fixed before mapping even begins: the choice of sensors tightly constrains what aspects of a performer's interaction can be captured digitally. Additionally, the limited real-time performance of common operating systems often requires real-time audio systems to incur significant latency to avoid dropouts in the audio stream.
This talk will begin by demonstrating that low latency and high interaction bandwidth are not abstract technical concerns: they are vital preconditions to making highly responsive digital musical instruments, as demonstrated in both qualitative and quantitative performer studies. We will then discuss how we have achieved these goals in Bela, an embedded computing platform for making richly interactive, low-latency musical systems. After 2 years of research, Bela was released to the public as an open-source community platform in 2016, and we continue to expand its ecosystem of hardware and software capabilities. The talk will reflect on lessons learned over that time which can be of use to other technology creators, especially those working with embedded hardware.
The final section of the talk will consider the aesthetic implications of engineering decisions, exploring how prioritising low latency, high bandwidth, and certain approaches to real-time signal processing can lead designers toward different patterns of thinking, ultimately producing different instruments than we would be likely to see with other tools. No tool can be aesthetically neutral, and we argue for designers to embrace the non-neutrality of tools and to engage in more critical reflection on the values embedded in each piece of technology.

SMC-22 Paper Session 1

Emulating Diode Circuits with Differentiable Wave Digital Filters

Jatin Chowdhury
Unaffiliated
jatin@ccrma.stanford.edu

Christopher Johann Clarke
Singapore University of Technology and Design
christopher_clarke@mymail.sutd.edu.sg

ABSTRACT

Wave Digital Filters and neural networks are two popular solutions for circuit modelling. This paper describes the development of a Differentiable Wave Digital Filters library. Diode clipper circuits were constructed. A dataset was collected from the circuits and, with the library, was used to train a real-time deployable model. The trained model has higher accuracy and similar computation time when compared to traditional white-box models.

1. INTRODUCTION

Virtual analog (VA) modelling is often divided into two non-distinct types of approaches. "White-box" modelling involves developing a circuit model based on the physical interactions of the circuit elements, while "black-box" modelling involves taking measurements from the circuit, and creating a digital system that replicates perceptually relevant aspects of the circuit's behaviour [1]. VA modelling approaches that combine elements of both white-box and black-box methods are typically referred to as "grey-box" approaches.

Wave Digital Filters (WDFs) are a white-box method that works by modelling individual circuit elements in the wave domain, and modelling the interactions of those elements with wave domain "adaptors" [2, 3]. WDFs are a powerful tool due to their modular and flexible nature; however, as with many other white-box approaches, they may provide inaccurate results when modelling circuits containing components that behave in a non-ideal manner.

In recent years, there has been significant research on developing black-box models using neural networks [4–6]. While this approach can achieve high levels of accuracy when comparing the model to the reference circuit, one drawback is that it can be difficult for neural network-based models to include circuit control parameters, particularly continuous controls such as potentiometers. Wright et al. suggest training neural networks using control values as additional inputs [6], however this approach often requires training a larger network, which requires more computing resources to run in real-time.

Copyright: © 2022 Jatin Chowdhury et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

One potential solution to the respective issues of the WDF and neural network modelling techniques is to combine them via Differentiable Digital Signal Processing (DDSP) [7]. The fundamental idea behind DDSP is to implement basic signal processing building blocks within a framework of automatic differentiation. Then, gradient descent may be used to optimize various parameters of the signal processing algorithm for a given set of input and target data. In recent years, DDSP has been applied to IIR filter design [8] and parameter discovery for white-box circuit models [9].

This paper proposes a grey-box modelling technique using Differentiable Wave Digital Filters (DWDFs). The DWDF technique involves constructing a WDF model of a reference circuit, and then training neural networks to replace one or more of the circuit elements in the WDF model. With this technique, non-ideal components can be modelled with a high degree of accuracy, since the neural networks may be trained with data collected from the actual circuit. Further, including the circuit's control parameters in the model is trivial, so long as the control parameters are connected to circuit elements being modelled with traditional WDF elements.

The structure of this paper is as follows: Section 2 discusses the development of a DWDF library, and the use of that library for solving simple parameter discovery tasks. Section 3 presents a process for training neural networks to emulate the behaviour of anti-parallel diodes in the wave domain. Section 4 considers the implementation of WDF models with neural network components for real-time use.

2. DIFFERENTIABLE WAVE DIGITAL FILTERS

While several WDF libraries exist [10, 11], they are primarily focused on implementing real-time circuit models, and are not well-suited for differentiation. With that in mind, a new WDF library was implemented in Python, using the TensorFlow framework for automatic differentiation [12]. The source code for the DWDF library is available on GitHub [13].

2.1 Library Implementation

Wave Digital Filters operate on wave variables, rather than the Kirchoff variables typically used for analyzing circuits (voltage v, and current i). The wave domain variables are

defined generically as,

$$\begin{aligned} a &= R_0^{\rho-1} v + R_0^{\rho} i \\ b &= R_0^{\rho-1} v - R_0^{\rho} i \end{aligned} \tag{1}$$

where a is defined as the incident wave for a given circuit port, b is the reflected wave, and R_0 is the port impedance. ρ is a wave definition parameter: "voltage waves" are defined for $\rho = 1$, while "current waves" are given when $\rho = 0$. In this writing, only voltage waves will be used.

2.1.1 DWDF 1-Ports

Most simple circuit elements, such as resistors and capacitors may be implemented as 1-port elements. A resistor is defined by the voltage wave relationship,

$$b = 0 \tag{2}$$

while a capacitor maybe similarly characterized as,

$$b = az^{-1} \tag{3}$$

where z^{-1} is defined as a 1-sample delay. Both of these circuit elements may be trivially implemented with TensorFlow, and may therefore be differentiated automatically. For cases where the resistance or capacitance of a given circuit element may not be known, the library also allows the component value to be initialised as a "trainable" variable.

2.1.2 DWDF Adaptors

Simple WDF adaptors such as series and parallel adaptors may be implemented generally as N-port elements. These adaptors are typically implemented as 3-port adaptors, since any N-port adaptor can be made up of a chain of 3-port adaptors. A series adaptor is defined by the voltage wave relationship,

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -1 \\ -R_1 & \frac{R_2}{R_1+R_2} & \frac{R_1}{R_1+R_2} \\ -R_2 & -\frac{R_2}{R_1+R_2} & \frac{R_1}{R_1+R_2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \tag{4}$$

where a_n, b_n, and R_n are the incident wave, reflected wave, and port impedance at a given port. A parallel adaptor is similarly defined by the voltage wave relationship.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0 & \frac{R_2}{R_1+R_2} & \frac{R_1}{R_1+R_2} \\ 1 & -\frac{R_1}{R_1+R_2} & \frac{R_1}{R_1+R_2} \\ 1 & \frac{R_2}{R_1+R_2} & -\frac{R_2}{R_1+R_2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \tag{5}$$

The DWDF library implements these adaptors using the one-multiply form described in [2]. When training a DWDF structure, TensorFlow will automatically propagate gradients through the relevant adapters so that a quantity anywhere in the structure maybe optimized via gradient descent.

2.2 Parameter Discovery with DWDFs

As a test of the DWDF models constructed with the library, two simple parameter discovery tasks were attempted, similar to those outlined in [9].

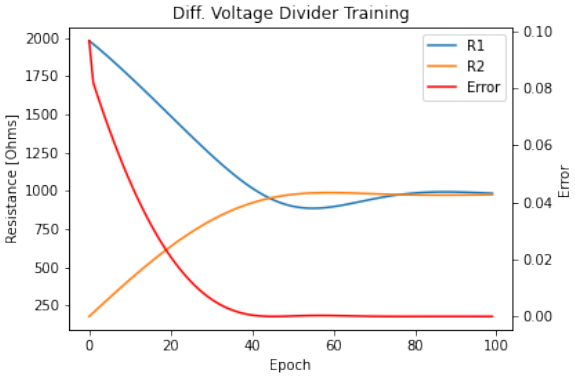


Figure 1: Training the voltage divider WDF model.

2.2.1 Voltage Divider

For the first task, synthetic data was generated for a simple voltage divider circuit made up of two equivalent resistors, corresponding to a gain of $G = 0.5$. A corresponding WDF model was constructed, using resistors with starting values of 2 kΩ and 100 Ω, both initialised as trainable variables. The WDF model was trained on the synthesized data for 100 epochs, using an Adam optimizer [14],

$$\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \cdot \frac{m_t}{\sqrt{v_t} + \hat{\epsilon}}, \tag{6}$$

where θ is the quantity being optimized, α is the initial learning rate, m_t is the exponential moving average of the gradient, and v_t is the squared gradient. Hyperparameters β_1 and β_2 represent the exponential decay rates of the first- and second-order moment estimates respectively, with default values $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For the voltage divider circuit, the Adam optimizer was given an initial learning of $\alpha = 25$ Ω. The model was trained using a mean-squared error (MSE) loss function,

$$L_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (y_t(i) - y_p(i))^2 \tag{7}$$

where y_t represents the "target" data, y_p represents the "predicted" signal output by the model, and N represents the length of the signal in samples. Table 1 shows the start and end values for each circuit element, as well as the final error. Fig. 1 plots the component values and error over the course of the training process. The component values after training correspond to a gain of $G = 0.502$.

2.2.2 RC Lowpass Filter

For the second task, synthetic data was generated for a first-order RC lowpass filter circuit, with a cutoff frequency of $f_c = 720$ Hz. A corresponding WDF model was constructed, with an initial cutoff frequency of 159 Hz ($R = 1$ kΩ, $C = 1$ μF), with both component values initialised as trainable variables. The model was again trained with an MSE loss function for 100 epochs. The resistor was trained with an Adam optimizer with an initial learning rate of $\alpha = 25$ Ω, while the capacitor was trained with a separate Adam optimizer with an initial learning rate of

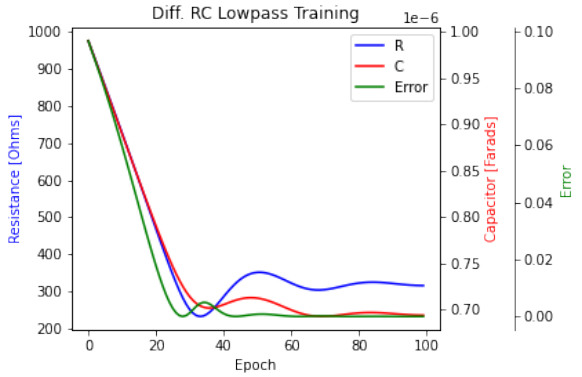


Figure 2: Training the RC lowpass WDF model.

Circuit	Element			Error [V]
Voltage Divider	R_1	Start	2 kΩ	3.4e−6
		End	984.75 Ω	
	R_2	Start	100 Ω	
		End	975.65 Ω	
RC Lowpass	R	Start	1 kΩ	2.58e−5
		End	315.99 Ω	
	C	Start	1 μF	
		End	693.8 nF	

Table 1: Training statistics for voltage divider and RC lowpass parameter discovery tasks.

$\alpha = 10$ nF. Table 1 shows the start and end values for each circuit element, as well as the final error. Fig. 2 plots the component values and error over the course of the training process. The component values after training correspond to a cutoff frequency of $f_c = 726$ Hz.

3. NEURAL WDF DIODE MODELS

While traditional wave domain diode models can achieve high accuracy when compared to the expected behaviour of ideal diodes, they typically require the evaluation of the Lambert $\mathcal{W}$ function at least once per-sample, which can limit the performance of real-time implementations [15]. As a result, real-time implementations often use lookup tables or approximations, which offer a trade-off between accuracy and performance [16]. Further, manufacturing inconsistencies and other real-world factors may cause diodes to behave non-ideally, thereby decreasing the accuracy of traditional wave domain models.

Neural networks offer a potential solution to these limitations, by leveraging data measured from a physical circuit to help train a more accurate model. As an example, this section will present a WDF model of an RC diode clipper (similar to one found in many guitar distortion pedals), in which the diodes are modelled using a neural network.

3.1 Diode Circuit Data

Data for the neural networks was prepared by constructing the diode clipper circuit shown in Fig. 3. This circuit varied with the amount of diodes on the "upward-facing" side

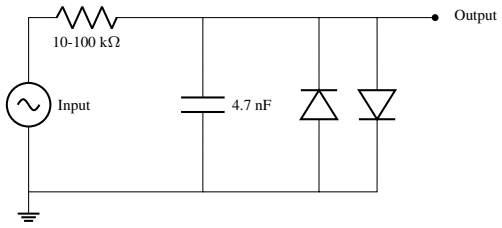


Figure 3: Diode Clipper with anti-parallel diode configuration. In this schematic, there are only 1 upward-facing and 1 downward-facing diode.

Diode Notation	Upward Diodes	Downwards Diodes
1U1D	1	1
1U2D	1	2
1U3D	1	3
2U2D	2	2
2U3D	2	3
3U3D	3	3

Table 2: This table shows the notation used for the labelling of the various diode clipper schematics, wherein 1U1D refers to a diode clipper with 1 upward-facing diode and 1 downward-facing diode.

and "downward-facing" side. Table 2 shows the different anti-parallel diode configurations sampled.

Following the construction of the 6 diode clipper circuits, a dataset was prepared. The input data for this dataset was taken from the IDMT-SMT-Guitar Dataset [17]. About 14 seconds of audio was used as input to the diode clipper circuit. The audio was output from a Universal Audio Apollo-Twin to the diode clipper circuit, and was sampled with a Digilent Analog Discovery 2 USB Oscilloscope, as shown in Fig. 4. The oscilloscope offers a range of sampling rates from 50 mHz to 100 MHz; measurements were made at 50 kHz since it was the closest option to a standard audio sampling rate. Each diode clipper circuit was sampled at 5 different resistor values (10 kΩ, 25 kΩ, 45 kΩ, 75 kΩ, 100 kΩ). The capacitor value remained unchanged.

Although 14 seconds seems like a small amount of data, one must note that model performance is being evaluated at the sample level, meaning that there will be about 4×10^6 samples to be used for training. 80% of the data was used for training the model, and 20% was used to validate the model's accuracy.

3.2 Diode Network Architecture

A network architecture was chosen as a sequence of fully-connected layers similar to the network presented in [18], with two inputs (the incident wave and port impedance) and one output (the reflected wave). Since diodes typically exhibit nonlinear behaviour, a $\tanh$ activation function is used in between each neural network layer. To improve training speed, the port impedance was replaced with the $\log$ of the port impedance, and the reflected wave was replaced with the negation of the reflected wave. The model

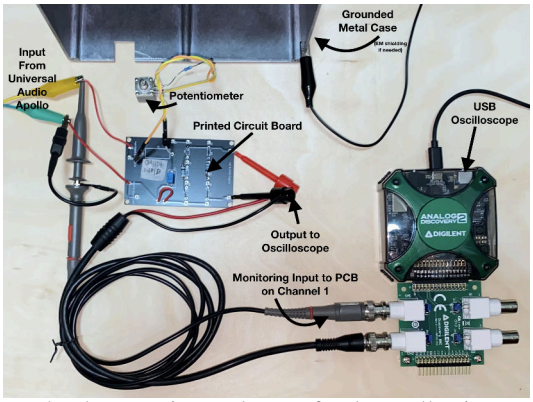


Figure 4: The experimental setup for data collection. Input comes from a Universal Audio Apollo-Twin audio interface, and data is captured and logged by a Digilent Analog Discovery USB Oscilloscope.

hyperparameters consist of the number of "hidden" layers to use in between the model inputs and outputs, as well as the size of the hidden layers. A visualization of an example "2x4" network with 2 hidden layers, each with 4 fully-connected units can be seen in Fig. 5. When trained, the diode network represents a memoryless mapping between the inputs and outputs.

$$b = -f\left(\begin{bmatrix} a \\ \log(R) \end{bmatrix}\right) \tag{8}$$

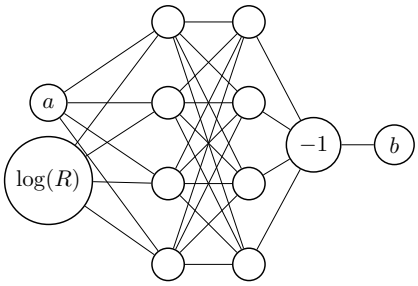


Figure 5: An example "2x4" diode network, with 2 hidden layers, each with 4 fully-connected units.

3.3 Diode Network Pre-Training

While it is expected that the diode network could be trained entirely within the DWDF model, training would be much slower than training the network outside of the DWDF model (due to the overhead introduced by the other WDF elements). With that in mind, each diode network was "pre-trained" against synthetic data, generated using wave domain diode equations derived from the Shockley diode law [15]. The training signal consists of a linear ramp of incident voltage waves ranging from $[-2.5, 2.5]$ V, repeated for exponentially increasing port impedances in the range $[10, 10^9]$ Ω (see Fig. 6).

Diode networks were pre-trained for 2000 epochs, using an Adam optimizer with a starting learning rate of 2e−5. The networks were trained with a combined loss function

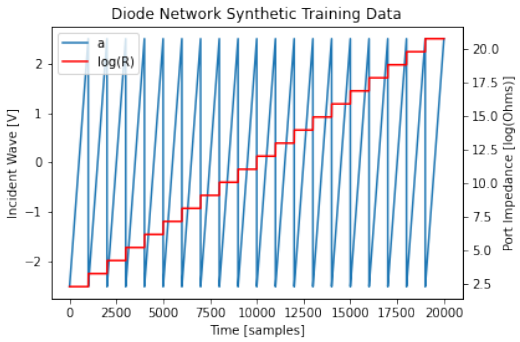


Figure 6: Synthetic data used for pre-training diode networks.

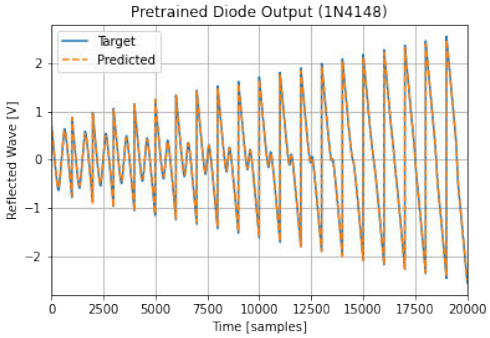


Figure 7: Pre-trained network results for 1-up/1-down 1N4148 diodes.

of mean-squared error, plus normalized error-to-signal ratio (ESR), defined as,

$$L_{\text{ESR}} = \sqrt{\frac{1}{N}\frac{\sum_{i=1}^{N}(y_t(i) - y_p(i))^2}{\sum_{i=1}^{N} y_t(i)^2}} \tag{9}$$

where y_t and y_p represent the "target" and "predicted" signal respectively, and N represents the length of the signal in samples. Then, from Equation (7).

$$L_{\text{TOT}} = L_{\text{MSE}} + L_{\text{ESR}} \tag{10}$$

Fig. 7 shows the results of pre-training a 2x8 network to model a set of 1-up/1-down 1N4148 diodes.

3.4 RC Diode Clipper Training

A DWDF model of the diode clipper circuit was constructed, with the diode set in the WDF model replaced by the neural network architecture shown above.

3.4.1 Training Hyperparameter Search

The DWDF models were trained using an Adam optimizer, as described in Equation (6) In order to determine the ideal training parameters for the differentiable diode clipper models, the training process was run for 100 epochs using the 1-up/1-down dataset with a 2x16 network, using a set of different training parameters for each run, as shown in Fig. 10. After completing the hyperparameter search, the set of hyperparameters resulting in the lowest loss values were: $\alpha =$ 1.0e-4, $\beta_1 = 0.5$, and $\beta_2 = 0.999$.

3.4.2 Determining Ideal Network Size

Next, the training process was run for 500 epochs using the 1-up/1-down dataset with a variety of network sizes, as shown in Table 3. From the results, it can be seen that using a "wider" network with larger hidden layers can improve the network accuracy more so than using a "deeper" network with more hidden layers. It was determined that the 2x16 network size should be used for training future networks, since it was able to achieve the highest accuracy.

Model	Pre-Training	Epoch 0	Epoch 100	Epoch 500
2x4	2.57e-3	3.22e-2	1.15e-2	9.31e-3
2x8	8.55e-4	3.01e-2	7.68e-3	5.96e-3
2x16	1.03e-4	1.14e-2	6.90e-3	4.42e-3
4x4	1.49e-3	2.13e-2	9.67e-3	7.88e-3
4x8	7.11e-4	2.11e-2	8.28e-3	4.92e-3

Table 3: Training results for 1-up/1-down models with different network sizes. "Pre-Training" shows the final loss values after pre-training. The final three columns show the validation loss values after epochs 0, 100, and 500.

3.4.3 Training Results

DWDF models of the diode clipper circuit containing 2x16 networks in place of the wave domain diode element were trained for 500 epochs for each diode configuration, using the training hyperparameters determined above. Table 4 shows the results of these training runs. The pre-training loss shown in the second column of Table 4 is the network error after the pre-training step described above. Note that the loss at Epoch 0 (third column) can be interpreted as the error between the ideal diode equations and the measured data. The neural network models were able to improve upon the initial error by more than a factor of two in almost all cases.

Finally, an additional 2x16 model was trained for 2000 epochs for the 1-up/1-down dataset, results in a final validation loss of 3.78e-3. Plots of the training and validation output signals before and after the training run can be seen in Fig. 9. From visual inspection, it appears that the largest discrepancies between the two signals occur during the extreme peaks in the signal. It is expected that adjustments to the loss function used to train the networks could improve the network performance for these parts of the signal.

Config	Pre-Training	Epoch 0	Epoch 100	Epoch 500
1U-1D	1.03e-4	1.14e-2	6.90e-3	4.42e-3
1U-2D	1.70e-4	2.36e-2	7.79e-3	6.12e-3
1U-3D	1.25e-4	2.77e-2	6.30e-3	5.05e-3
2U-2D	1.71e-4	1.38e-2	8.12e-3	7.45e-3
2U-3D	1.01e-4	1.89e-2	8.66e-3	7.04e-3
3U-3D	3.07e-4	1.25e-2	8.25e-3	6.04e-3

Table 4: Training results for 2x16 networks with different diode configurations. "Pre-Training" shows the final loss values after pre-training. The final three columns show the validation loss values after epochs 0, 100, and 500.

Another useful comparison is to examine the transconductance of the neural network diode model compared to the transconductance characteristic predicted by the Shockley diode law [19], shown in Fig. 8. From the asymmetry in the transconductance of the neural model, it is likely that the upward- and downward-facing diodes used in the circuit did not have identical characteristics, as the ideal model assumes. Further, the loss in current at higher voltages indicates that the diodes may have exhibited some internal resistance that is not accounted for by the ideal model.

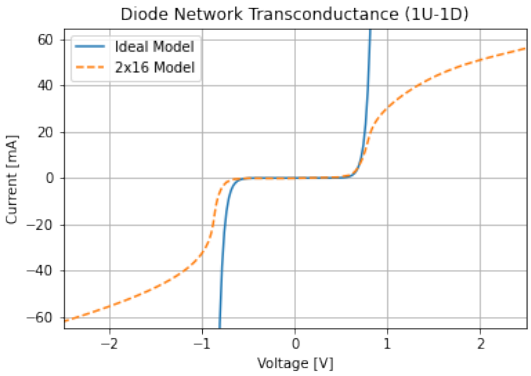


Figure 8: Comparing the transconductance characteristic between the ideal Shockley diode law, and the 2x16 neural network for the 1-up/1-down diode configuration.

4. REAL-TIME CONSIDERATIONS

VA models are often implemented as part of a real-time system that may be used for sound mixing/mastering, or musical performance. With that end in mind, an audio plugin was developed containing an implementation of the 1-up/1-down WDF diode clipper model using several different methods for modelling the wave domain diodes. The first implementation uses the wave domain diode equations presented in [15], along with a high-precision C++ implementation of the Wright Omega function in order to evaluate the Lambert $\mathcal{W}$ function [20]. The second implementation uses the same diode equations, this time using an approximate implementation of the Wright Omega function [16]. The remaining implementations use the trained diode models developed in the previous section, implemented using the RTNeural library for performing neural network inferencing in real-time [21]. Source code for the plugin is available on GitHub [13].

4.1 Performance Comparison

In order to compare the performance of the different diode models, a performance benchmark was developed using the Google Benchmark library. [1] . The benchmark initialises the diode clipper WDF with the given diode model, and processes 100 milliseconds of audio at a sample rate of 96 kHz. This process will repeat until the benchmarks time

[1] `https://github.com/google/benchmark`

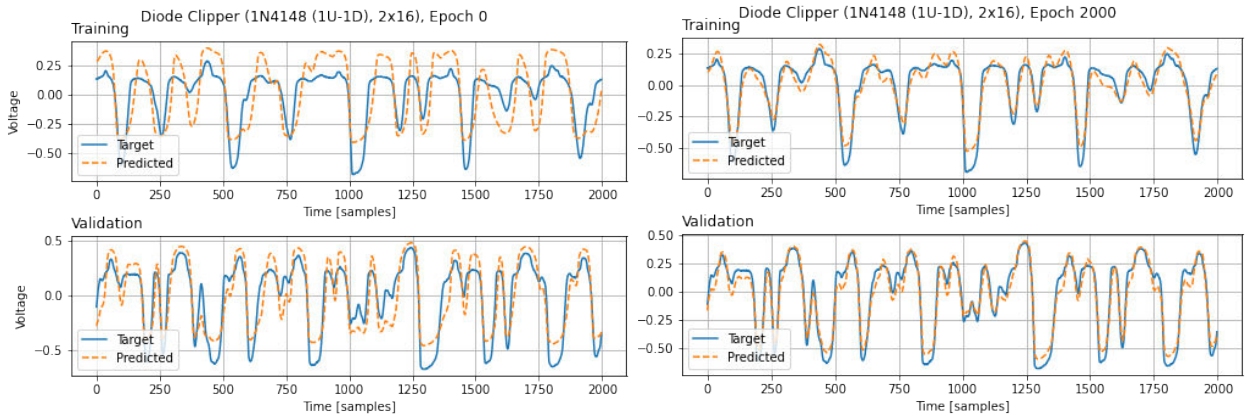


Figure 9: Before and after comparison of the training process for the 2x16 model.

Model	# Iterations	x Ideal
Ideal Model	1816	–
Approx. Model	20779	11.44
2x4 Model	7006	3.86
2x8 Model	4401	2.42
2x16 Model	2302	1.27
4x2 Model	3531	1.94
4x8 Model	2903	1.60

Table 5: Results of the diode clipper performance benchmarks. "Ideal/Approx. Model" refers to the diode model implemented with a high-precision/approximate Wright Omega function. The second column shows how many iterations the benchmark was able to perform within 5 seconds. The third column shows how many times faster each model is when compared to the ideal model.

out after 5 seconds. The number of iterations completed within 5 seconds can then be used as a "score" to compare the run-time performance between the models. The benchmarks were run on a 2018 Mac Mini, with a 3.2 GHz Intel Core i7 CPU. Table 5 shows the results of the performance benchmarks, including the number of iterations completed within 5 seconds, as well as the number of iterations compared against the ideal model score. The results of the benchmark show that the neural network models can out-perform the high-precision implementation, although all the neural network models are clearly out-performed by the approximate model. For the purposes of practical implementations of diode clipper circuit models, the implementer should choose between the speed of using a model based on mathematical approximations of lookup tables relative to the improved accuracy given by the neural network model.

5. CONCLUSION

This paper has outlined the development of a Differentiable Wave Digital Filter library, that can be used to train neural network models of circuit components via gradient descent. The DWDF library has been used to train neural network models of anti-parallel diodes, which may be used in models of audio circuits. From a "white-box" perspective, DWDFs offer the ability to augment wave digital circuit models with data measured from physical circuits. From a "black-box" perspective, DWDFs offer a method to construct neural network circuit models that is modular and utilizes prior knowledge about the circuit. The WDF models constructed with trained neural networks can be implemented for real-time use with comparable performance to a model constructed with traditional WDF elements.

Future research in this area will focus on extending the scope of DWDF models, to include models of circuits with more complex topologies, such as circuits with multi-port elements including tubes, transistors, and op-amps. In particular, training differentiable models of $\mathcal{R}$-type adaptors [3] could offer many possibilities for developing data-driven models of more complicated circuits. In particular, WDF models of circuits containing multi-port nonlinearities often require large multi-dimensional lookup tables or computationally expensive iterative solvers. Replacing these multi-port nonlinearities with neural networks has the potential to improve the real-time performance of these circuit models.

DWDFs also make it possible for neural models of circuit components that were originally trained in one circuit to be used in a WDF model of a completely separate circuit. Exploring this possibility is an interesting topic for future research.

Another potentially interesting line of study is the use of machine learning to generate a WDF topology for an unknown circuit. While the DWDF strategy presented here would be useful for optimizing the generated topologies, the topology generation itself would require an approach that is not based on gradient descent, such as genetic algorithms or heuristics.

Acknowledgments

The authors would like to thank the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University for providing computing resources necessary for this research. Thanks also to Camille Noufi, Dirk Roosenburg, and Champ Darabundit for useful conversations about neural network training strategies.

6. REFERENCES

[1] F. Germain, "Non-Oversampled Physical Modeling for Virtual Analog Simulations," Ph.D. dissertation, Stanford University, June 2019. [Online]. Available: https://searchworks.stanford.edu/view/13250111

[2] A. Fettweis, "Wave Digital Filters: Theory and Practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, Feb. 1986.

[3] K. J. Werner, "Virtual Analog Modeling of Audio Circuitry Using Wave Digital Filters," Ph.D. dissertation, Stanford Univeristy, June 2016. [Online]. Available: https://searchworks.stanford.edu/view/11891203

[4] M. A. Martínez Ramirez and J. D. Reiss, "Modeling of Nonlinear Audio Effects with End-to-End Deep Neural Networks," *arXiv e-prints*, p. arXiv:1810.06603, Oct. 2018.

[5] E. Damskägg, L. Juvela, and V. Välimäki, "Real-Time Modeling of Audio Distortion Circuits with Deep Learning," in *Proc. of the 16th Sound and Music Computing Conference (SMC-2019)*, May 2019.

[6] A. Wright, E. Damskägg, and V. Välimäki, "Real-Time Black-Box Modelling with Recurrent Neural Networks," in *Proc. of the 22nd Int. Conference on Digital Audio Effects (DAFx-19)*, Sept. 2019.

[7] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=B1x1ma4tDr

[8] B. Kusnetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR Filters for Machine Learning Applications," in *Proc. of the 23rd Int. Conference on Digital Audio Effects (DAFx-20)*, Sept. 2020.

[9] F. Esqueda, B. Kusnetsov, and J. D. Parker, "Differentiable White-Box Virtual Analog Modelling," in *Proc. of the 24th Int. Conference on Digital Audio Effects (DAFx-21)*, Sept. 2021.

[10] D. Roosenburg, E. Stine, R. Michon, and J. Chowdhury, "A Wave-Digital Modeling Library for the Faust Programming Language," in *18th Sound and Music Computing Conference (SMC-2021)*, June 2021.

[11] M. Rest, R. Dunkel, K. J. Werner, and J. Smith, "RT-WDF—A Modular Wave Digital Filter Library with Support for Arbitrary Topologies and Multiple Nonlinearities," in *Proc. of the 19th Int. Conference on Digital Audio Effects (DAFx-16)*, Sept. 2016, p. 287–294.

[12] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[13] J. Chowdhury and C. J. Clarke, "Differentiable WDFs," https://github.com/jatinchowdhury18/differentiable-wdfs, 2022.

[14] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2015.

[15] K. J. Werner, V. Nangia, A. Bernardini, J. Smith, and A. Sarti, "An Improved and Generalized Diode Clipper Model for Wave Digital Filters," *Journal of the Audio Engineering Society*, Oct. 2015.

[16] S. D'Angelo, L. Gabrielli, and L. Turchet, "Fast Approximation of the Lambert W Function for Virtual Analog Modelling," in *Proc. of the 22nd Int. Conference on Digital Audio Effects (DAFx-19)*, Sept. 2019.

[17] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score- and Instrument-Related Parameters." in *DAFx*, 2014, pp. 219–226.

[18] J. D. Parker, F. Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. of the 22nd Int. Conference on Digital Audio Effects (DAFx-19)*, Sept. 2019.

[19] W. Shockley, "The Theory of P-N Junctions in Semiconductors and P-N Junction Transistors," *The Bell System Technical Journal*, vol. 28, no. 3, pp. 435–489, 1949.

[20] P. W. Lawrence, R. M. Corless, and D. J. Jeffrey, "Algorithm 917: Complex Double-Precision Evaluation of the Wright ω Function," *ACM Trans. Math. Softw.*, vol. 38, no. 3, Apr. 2012. [Online]. Available: https://doi.org/10.1145/2168773.2168779

[21] J. Chowdhury, "RTNeural: Fast Neural Inferencing for Real-Time Systems," 2021. [Online]. Available: https://arxiv.org/pdf/2106.03037.pdf

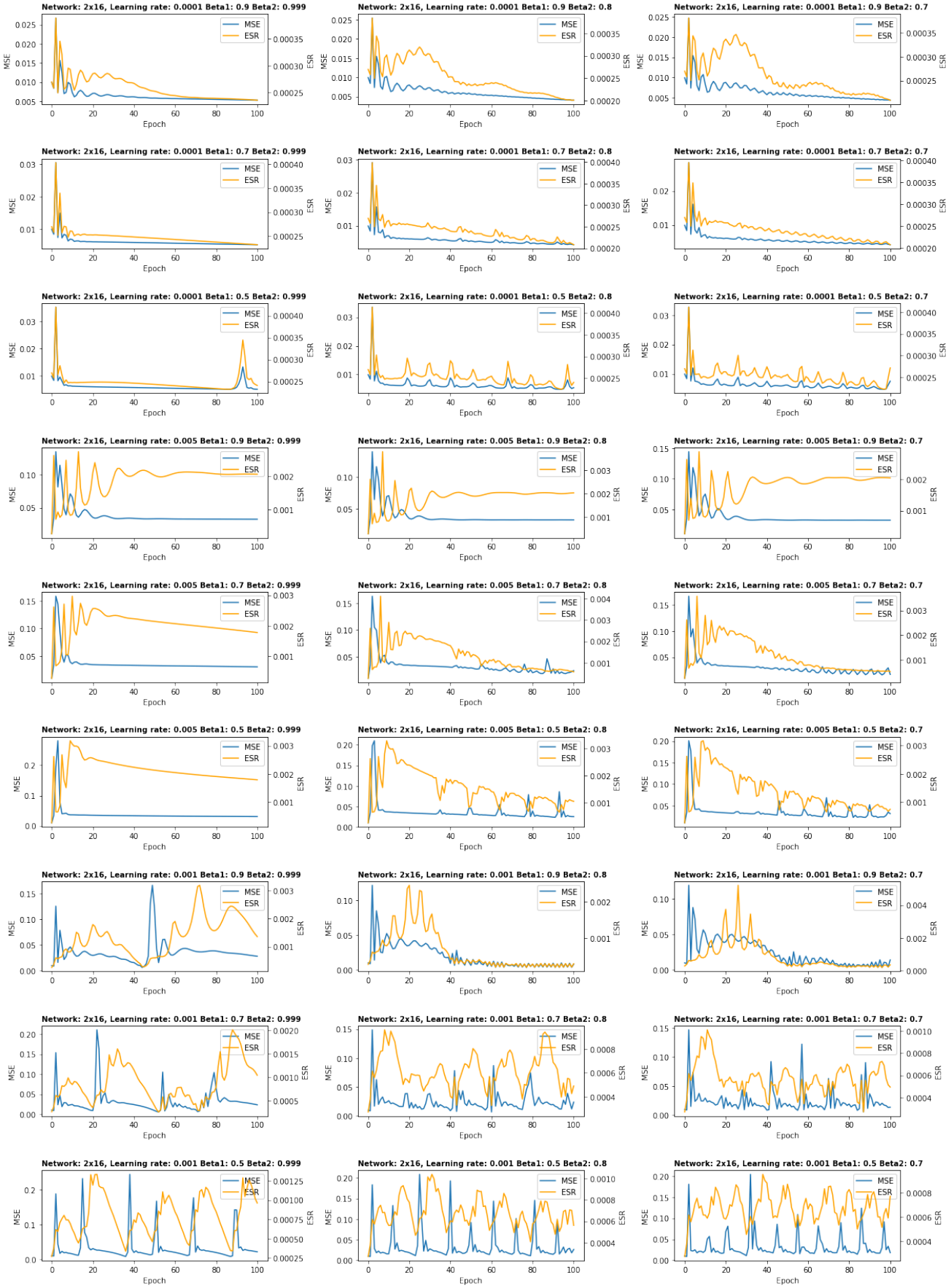


Figure 10: Results of network hyperparameter search.

Approaching Spatial Audio Processing by Means of Decorrelation and Ring Modulation in Ambisonics

Paul Goutmann
ArTeC
Université Paris 8
Musidanse/CICM
paul@goutmann.net

Alain Bonardi
Université Paris 8
Musidanse/CICM
alain.bonardi@univ-paris8.fr

ABSTRACT

This paper presents an approach to spatial audio processing in ambisonics as a part of musical practice. We present the control of two treatments - the decorrelation and ring modulation in ambisonic - to express how the design of digital objects feeds into the composition of spatial attributes of sound and the musical idea of space. The decorrelation factor gives the possibilities rangy from localizable sources to diffuse sound field, and the modulation factor enables the variation between static situations to complex movement in space. These parameters encapsulate a multitude of operations on the signal as well as ways to think about interactions with spatial attributes of sound. Based on an experimental methodology of research-creation combining audio experiments with development, this approach shows how rich it is to explore different layers of our compositional environments to propose new paradigms of spatial audio. This has resulted in a new version of the decorrelation and ring modulation inside the Faust library *abc*.

1. INTRODUCTION

The spatial dimension of sound is essential in the field of contemporary music creation, especially in electroacoustic and electronic music. Today, concert halls and studios are equipped with multipoint diffusion systems. Since the 1990s, software implementing sound spatialization models have been created and shared by the community of compositors, computer music designers and audio engineers [1]. The development of these technologies feeds new paradigms of representation and ways to think about space in music [2]. Most of the spatialization techniques are based on point sources in close relationship with spatial hearing models [3] [4] [5]. Accessible musical operations for creators are the placement of sources, designing movements and room acoustic simulation. However, other approaches of interaction with spatial attributes of sound exist [6] [7] [8]. The project *HOALibrary* contributed to those approaches in synergy with many years of research and creation about the spatiality of sound at CICM. The *Centre d'Informatique et de Création Musicale* created by Horacio Vaggione in 1992 accommodates researches which have interactions [1] between creation, technologies and computer sciences especially spatialization since the beginning of the 2000s [9]. Developed by J. Colafrancesco, P. Guillot and E. Paris between 2012 and 2015 in the context of the Labex Arts H2H projects, *HOALibrary* is a library of spatial audio processing tools in ambisonics [10]. One of the particularities of this library for Max and Pd is the insertion of signal processes (well known in electroacoustic and mixed music) such as granulation, ring modulation or decorrelation into the ambisonic model. Unfortunately the Pd version is obsolete since 2019. In order to formalize twenty years of work on mixed music and to address the obsolescence of *HOALibrary*, Bonardi began to transcribe treatments of the object *hoa.process∼* in Faust (as well as ambisonic basics and multichannel processes) grouped in the library *abc.lib* [11] [2]. In this context, we began new developments and fine-tuning around the decorrelation and the ring modulator in ambisonics lean on a research-creation practice [3]. The musical interest of those two treatments is in part linked to the fact that they allow complex sensitive results through the manipulation of just one parameter, the factor. The decorrelation produces progressively diffuse sound fields from one mono source (Sound 6.1) and the ring modulation creates the sensation of complex movement from one mono source (Sound 6.2). We refer to sound examples all along the article which are indexed in the appendix (6) and available on the platform *HAL* of the *Centre pour la Communication Scientifique* of the CNRS.

Copyright: © 2022 Paul Goutmann et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[1] The interactions take place in research projects where the activity of creation permits to test in sensitive and qualitative ways the results of audio digital processes. https://cicm.mshparisnord.fr Accessed on: April 6, 2022

[2] The library is currently compiled for Max and Pd and available here with the Faust code: https://github.com/alainbonardi/abclib Accessed on: April 6, 2022

[3] A state of the art of 'research-creation' or 'artistic research' is beyond the scope of this paper. 'Research-creation' refers to a large set of practices and methods. For this paper, its doing research in interaction with creation in an academic context. Both part emerges from and merges with the other. Composing is here a manner to handle tools (theoretical and technological) and to produce knowledge in music, musicology and computer music sciences. Composing as a research activity is at the core of CICM and ArTeC. We explain in fact the methodology in section 3.

2. INTERACTING WITH SPATIALITY

In the practices of electroacoustic, mixed and electronic music, the work on the spatial dimension of sound cannot be reduced to spatialization [12] [13] [14] [15] [4] . Designing treatments as a way to design instruments is at the core of musical practices in digital environments. The *HOALibrary* project approached space in terms of spatial process of sound. The library has been conceived to give access to musicians to the operations in the framework of the HOA (High Order Ambisonic) model. In this section we will first present the way to create treatments in *HOALibrary* and their transcriptions in the Faust language. Then we will present the design of macro parameters as controllers for digital processes. Finally, we will go further and expose how the design of functions is an important step to give access to operations.

2.1 Spatial Processing

The B-Format proposed by M.Gerzon and HOA proposed by J.Daniel are techniques to record, synthesize, process and reproduce acoustic fields. These techniques are based on three steps: encoding, optimisation and decoding. The encoding principle is based on the description of the spherical functions (in 3D) or circular functions (in 2D) corresponding to the acoustic field at one point of space using an ensemble of reference functions. These reference functions are called 'harmonics', their number depends on the encoding order. The more harmonics are used, the more precise the encoding will be. In a 2D encoding there are $2(N+1)$ harmonics and in 3D encoding there are $(N+1)^2$ harmonics. They are organized in order m (line) and degree l (row) as $-l \leq m \leq +l$ and can be represented as in the Figure 1.

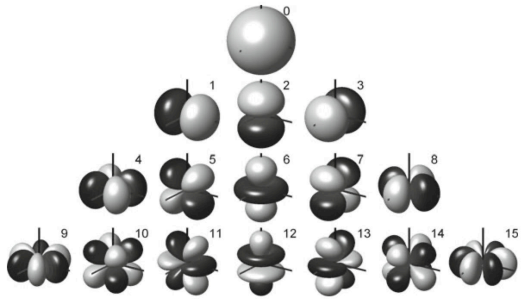


Figure 1. Spherical harmonics indexed by the ACN (*Ambisonic Channel Number*) in [16] .

The decoding stage consists of calculating the projection of the spherical function on the loudspeaker setup. In other words, calculating the gain sent to each speaker depending of a diffusion system which needs to include more speakers (set on a circle or a dome) than the number of harmonics. Processes in *HOALibrary* consist of applying digital audio treatment to each harmonic. Inspired by the object *poly*$\sim$, the objects *hoa.2d.process*$\sim$ or *hoa.3d.process*$\sim$ - takes as first argument the order of encoding - instances a patch of treatment for each harmonic [17]. For example, the object *hoa.2d.process*$\sim$ in the Figure 2, instances the patch *hoa.fx.decorrelation*$\sim$ (as in Figure 3 for Y_0) fifteen times corresponding to the numbers of harmonics in HOA 2D 7^{th} order. The process parameters are generally proportional to an harmonic coefficient which depend on the order, the degree and the number of harmonics. Since 2020 patches instantiated by the *hoa.process*$\sim$ are grouped in *abc.lib* in Faust. This functional language for signal processing gives the opportunity to clear operations especially with the primitive function 'par' which creates indexed operations in parallel. Multiple windows encapsulated in Max or Pd correspond to a few lines in Faust that can be edited using an embedded compiler.

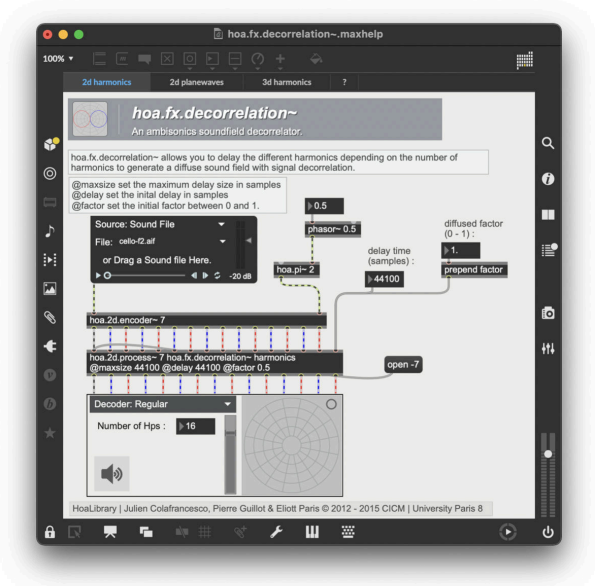


Figure 2. Help patch of the hoa.fx.decorrelation$\sim$ spatial process.

2.2 The Factor

Designing treatments is a way to build a part of our 'composable space'. This notion originally developped by Horacio Vaggione is the environment allowing composers to generate and control sound morphologies. Composable spaces are multidimensional and are themselves morphologies that can be designed [14]. The composable space offers controllers to operate on and compose sound. In *HOALibrary* and *abc.lib*, the controllers which are designed for decorrelation and ring modulation are accessible parameters, and notably the 'factor'. The 'factor' parameter is a meta-parameter which controls the depth of the treatment. The idea of Guillot and Paris was to begin to process higher harmonics for low values of the 'factor' up to the harmonic 0 for the maximum of the 'factor'. When these treatments are applied after a mono source encoding, the result can be described by a transition from point source to a diffuse sound field with a color chart behind

[4] These four papers are written by composers (except the last) which cover electroacoustic, mixed music, electronic music and musical installation. They are focused on their own approach to the spatial dimension of sound, and gives an idea of the range of approaches.

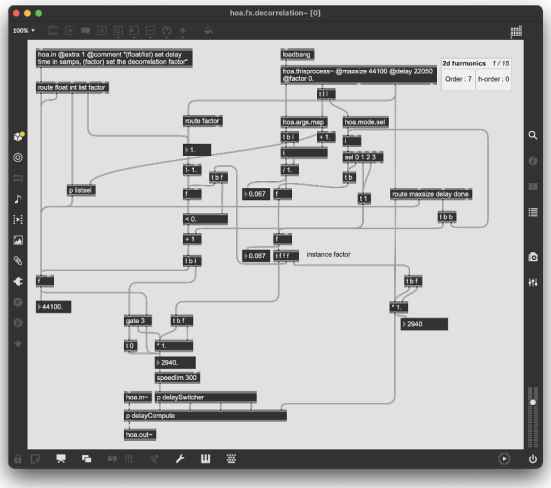


Figure 3. hoa.fx.decorrelation∼ patch.

these two states for the decorrelation [18] and a transition from point static situations to larger source and complex movements for the ring modulation.

2.3 Accessing Controls in Faust

One of the advantages of the Faust language is the possibility to define a user interface which will be automatically rendered at compilation. Primitives like buttons, sliders, and number boxes are generated. These primitives are parameterized when they are declared: initial value, minimum, maximum, interval. For example in the function $abc_2d_fx_decorrelation_ui(n)$ presented in the Listing 1, the function *fxDecorrelation* is called with four parameters which are the variables 'delay', 'factor', 'fdbk' and 'functiontype'. Those variables take the value of the associated primitive 'hslider' (horizontal slider).

```
abc_2d_fx_decorrelation_ui(n) = fxDecorrelation(
    n, delay, winfreq, factor, fdbk,
    functiontype)
  with {
    delay = hslider("v:synfxdecorrelation/delay
    [unit:samples]", 48000, 10, 262144, 1);
    window = 10; //by default a window of 10
    milliseconds for the interpolation of delays
    //
    winfreq = 1000. / window;
    factor = hslider("v:synfxdecorrelation/
    factor", 0, 0, 1, 0.001);
    fdbk = hslider("v:synfxdecorrelation/fdbk",
    0, 0, 0.99, 0.001);
    functiontype = hslider("v:synfxdecorrelation
    /functiontype", 0, 0, 21, 1);
    //
};
```

Listing 1. example of GUI with Faust in the abc.lib

If the function $abc_2d_fx_decorrelation_ui(n)$ is called in the 'process', at compilation the program will include a horizontal slider labeled 'delay', 'factor', 'fdbk' and 'functiontype' corresponding to the string. In *abc.lib*, graphic user interfaces are dissociated from audio process functions (the *fxDecorrelation* function in this example). Already at this step, the controllers are defined. For example the decorrelation is based on delay lines which can be feedback: should we leave the possibility to control the amount of feedback in the interval [0, 1] or [0, 1[? In the same way, if the interval for the factor is 0.1 or 0.001, the perceived result will not be the same. We will see in the next section that the controllers of the composable space are already designed in the function implementation.

3. DESIGNING DIGITAL OBJECTS

Interactions between composers, performers or computer music designers and sound qualities when computers are involved are multilayered. Building setups of hardware controllers, sensors or GUI is common in the computer music practice as well as the building of audio processors with such high level graphical environments as Pd, MaxMsp or Max4Live. Thinking about the interactions between all the layers of musical environments and the sonic user feedback is essential to understand the music made with these environments. The term 'digital object' used in the title encapsulates all these layers. In this section, we highlight a methodology of experimental practice to build treatments in a musical way. With the new developments and fine tuning on decorrelation and ring modulation we want to show the multi-scale aspect of designing digital objects that gather: implementation, design of the controllers, GUI and activation by the user. Our method is based on an iterative prototyping loop always linked to listening and composition: we develop audio processes with musical goals, tested them in the studio and used them in compositions. These studies tools are handled in works of Musicology Bachelor students (Paris 8 University) as well as in experienced composers' works. The results of the interactions between developments, the self used, feedback from students and from experienced composers are not quantitative but observed in a qualitative way [5]. All the tests were done in HOA 2D 3^{th} on 8 speaker (Genelec 8030A) portion of a 16 speaker dome in the studio of the CICM at the MSH PN and example are grouped in the appendix.

3.1 Decorrelation

The terms correlation and decorrelation refer to mathematics tools used in digital audio processing among other disciplines for musical purposes and for analyzing perception of space. We can think of correlation as similarity. For example, the cross-correlation function is a way to calculate the similarity between two signals. The decorrelation in digital audio applied to musical purpose consists in dephasing signals or applying all-pass filters. Extending

[5] We saw that this tools affect the students' approach of composition. Moreover the decorrelation takes a large place in the Anne Sèdes works. She documented her approach of decorrelation in ambisonics in [19] and the last version of the ambisonics decorrelation was implemented and tested in her last work *Écoute/Expension* for the choreograph Kitsou Dubois. Plus, as a part of his PhD research, Goutmann is working on an electroacoustic piece which handle the decorrelation in ambisonic *Compression/Dilatation* in the examples is a binaural version of the first part of this piece originally for a 3D setup of speakers (Sound 6.6). The piece was performed at the *Les visites insolites du CNRS* (October 4 2021) and during an electroacoustic concert *Séance d'écoute* (March 4 2022) at the MSH PN *Maison des Sciences de l'Homme Paris Nord*

the works of Kendall and Vaggione [20] [21], the decorrelation in *HOALibrary* (and then in *abc.lib*) is based on the dephasing of the harmonics after the encoding process with the insertion of delay lines. Inserting some delays between harmonics breaks the correlation of post-encoding signals and gives a rich perceptive result of a diffuse sound field. The decorrelation has initially two parameters: 'delay', 'factor'. We add two more: type of distribution of the delays ('functiontype') and feedback ('fdbk'). The factor determines how many harmonics will be delayed, the delay is the maximum delay time applied to the harmonics, the distribution of delay is the function used to spread the delays between the harmonics. The decorrelation is defined for each spatial component among:

$$P = \begin{cases} 2.N + 1 & \text{in 2D,} \\ (N+1)^2 & \text{in 3D} \end{cases} \quad (1)$$

where P is the number of harmonics at the ambisonic order N, by the delay d_i:

$$d_i = \begin{cases} 0 & \text{if } fa < 1 - \frac{i+1}{P}, \\ delay.f(\frac{i+1}{P}) & \text{else} \end{cases} \quad (2)$$

where $f(x)$ is the function of distribution called abc_{th} in the library. The blockdiagram of the decorrelation in Figure 4 shows that we hear the signal delayed or not according to the factor and the index of the harmonic, thanks to the two envelopes env1 and env1c - when one is equal to zero the other is equal to one.

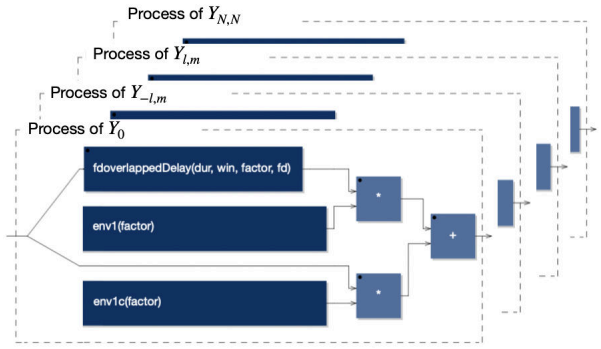


Figure 4. Blockdiagram of the decorrelation in *abc.lib*

3.1.1 *The stairs problem*

In this situation, if the factor value is in the interval [0, $\frac{1}{P}$] the last harmonic is delayed with the maximum delay time and all the others won't be delayed. Therefore, as the factor increases, the second to last harmonic will be delayed starting from a factor equal to $\frac{2}{P}$, the third to last will be delayed starting from $\frac{3}{P}$, *etc.* The sensitive result isn't a progressive generation of a diffuse sound field but a progressive insertion of static delay lines (Sound 6.3). Figure 5 illustrates the coefficient of delay time applied to each harmonic when the factor increases in 2D 3^{th} order: when the factor increases we first hear the last harmonic delayed ($Y_{-3,3}$), then second to the last ($Y_{3,3}$) and this until the first (Y_0).

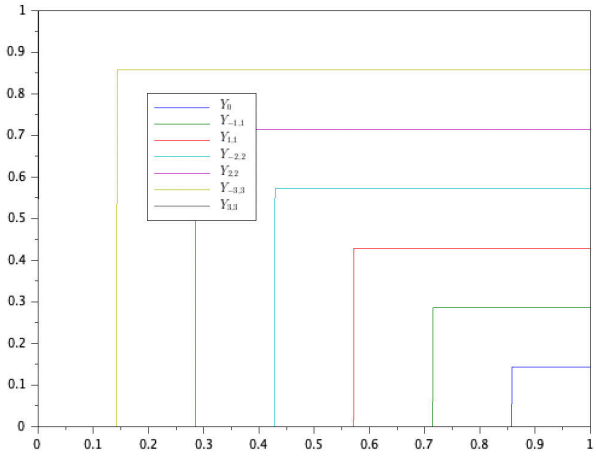


Figure 5. Plot of the delay coefficient per harmonic versus the factor.

In consequence, we decided to introduce the factor in the delay equation of each harmonic. That leads to the reduction to the gap between the delays of harmonics already delayed and harmonics that begin to be delayed when the factor increases. Moreover, the delays progressively increase as the factor increases as illustrated in the Figure 6 and that amplifies the effect of progressive generation of diffuse sound field (Sound 6.4). We have chosen to duplicate each signal corresponding to the harmonics and add a gate as a binary wet/dry controlled by the factor. The delay time is equal to 0 when the factor is under a threshold and the feedback introduces coloration and saturation. So when the factor is under the threshold we hear the signal without delay line and when the factor is higher the threshold we hear the signal with a delay line.

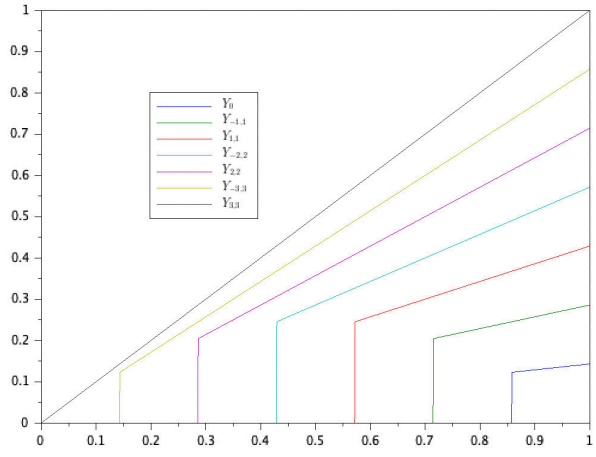


Figure 6. Plot of the delay coefficient per harmonic versus the factor when including the factor in the delay equation.

3.1.2 *The distribution choice*

The calculation of the delay time for each harmonic depends on the factor, the maximum delay time and the coefficient of the harmonic. The higher the order of the harmonic is, the deeper the treatment. But the delay time

applied if the factor is higher than the threshold increases linearly from the first harmonic to the last. Based on experiments on graphic user interfaces to control the decorrelation [18], we proposed to control the type of distribution with 21 functions called abc_{th} that are in the range [0, 1] when x ∈ [0, 1] (Sound 6.5).

3.2 Ring Modulation

Ring modulation consists in modulating a bipolar signal by another bipolar signal. In ambisonics, ring modulation consists in modulating each harmonic with a sinusoidal signal and has two parameters: the factor of modulation and the frequency of the carrier. If the factor is greater than the threshold:

$$\frac{(P-i-1)}{P} \tag{3}$$

the i_{th} ring modulator is active with a carrier c of:

$$c = f0.\frac{(i+1)}{P} \tag{4}$$

where P is the number of harmonics. If the factor is under the threshold, the original signal is provided. Therefore, ring modulators are progressively revealed when the factor increases. For example, after an encoding at third order in 2D, there are seven harmonics. The harmonic 0 will have a modulation with a carrier of $\frac{f_0}{7}$ and will be modulated between a factor of $[\frac{6}{7}, 1]$. Continuing the example, if the frequency of modulation is 10Hz the frequency of modulation of each harmonic will take the value shown in the Table 1 (Sound 6.2).

Y_{ith}	$freq_{mod}$
Y_0	1.42 Hz
$Y_{-1,1}$	2.85 Hz
$Y_{1,1}$	4.28 Hz
$Y_{-2,2}$	5.71 Hz
$Y_{2,2}$	7.14 Hz
$Y_{-3,3}$	8.57 Hz
$Y_{3,3}$	10.0 Hz

Table 1. Modulation frequency per harmonic for 2D 3^{th} order

The sensitive result of this process ranges from complex movements to timbre modification thanks to the behaviour of the ring modulation with a frequency modulation higher than 20 Hz.

3.3 FX/SYN

There are two modes of applying in *abc.lib*: FX and SYN. FX mode applies the process to the harmonics post-encoding. The idea is to be able to control the depth of the treatment on the sound field from a directional sound field to a diffuse sound field. The idea of the SYN mode is that it doesn't matter if the signal is not ambisonic encoded before applying the treatment because the process distorts the coherence of the intermediate representation of space. SYN mode is thought to synthesize directly a diffuse sound field from one mono source. But when the factor is equal to zero, the same signal is sent to all the decoder inputs as if a localized sound field was synthesized the source is localized in one direction [6]: 45°. For example, in the first order case, if the same signal is sent to all the entries of a decoder it get close to an encoding of a source at 45° (when the source angle is 45°, the output of $Y_{-1,1}$ and $Y_{1,1}$ are 0.707) and the intermediate representation of space (circular function) has shown in the Figure 7.

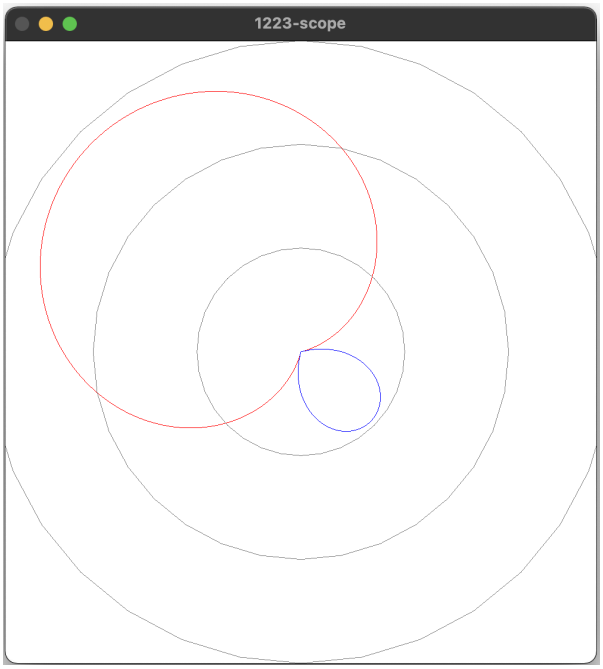


Figure 7. Intermediate representation of space with Y_0, $Y_{-1,1}$ and $Y_{1,1}$ equal to one. In red the positive and in blue the negative part of the function.

Consequently, we decided to cut all harmonics except the first when the factor is equal to 0. The objective was to create a strong mono when the factor is equal to zero and to go progressively to a diffuse sound field. We apply gain compensation and progressively fade in the other harmonics when the factor increases between $[0, \frac{1}{n}]$. The Figure 8 shows the gain compensation of the Y_0 per ambisonic order (until 7) versus the factor.

4. CONCLUSION

In this article we highlighted the potential of 'treatment' oriented approach for spatial audio processing in *HOALibrary* and *abc.lib*. We presented the richness of a research-creation methodology based on an iterative prototyping loop guided by listening with two examples: decorrelation and ring modulation in ambisonics. In the part 2 'Interacting with spatiality' we first provide a summary on ambisonics and the approach of space in *HOALibrary*. In the section 2.2, we presented a new paradigm to compose and

[6] In Head-Related Coordinate System, and the angle is given in anti-clockwise

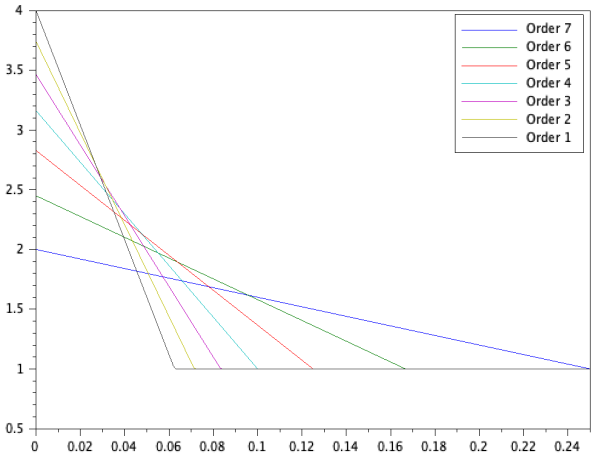


Figure 8. gain compensation per ambisonic order versus the factor

control spatial attributes of sound based on the design of macro-parameters. We then suggested that attention given to the access to the functions (the controllers) take a part in the design of that Vaggione calls our 'composable space'. Based on two examples, we explained our methodology in part 3 'Designing digital objects'. First, we presented in the section 3.1 the fine tunings on the delay time equation, then in the section 3.1.2 the different types of distributions between the delay lines that we use in the library. We presented in the section 3.2 the ring modulation and a way to distribute all the frequencies of modulation to the spatial harmonics. Finally, we presented two modes of treatment in *HOALibrary* and *abc.lib*: FX and SYN ; we explain our choice to use only the first spatial component when the factor is equal to zero for the SYN mode. In future work, we plan to investigate other treatments of the library [7] using the same methodology, combining tests and musical experiments to improve and tune the parameters/meta-parameters of the processes. Furthermore, we will investigate the particularity of the ambisonic situation in different multi-channel situations. If processes of *abc.lib* are applied to the signals associated with harmonics or other multi-channel signals, it will be the same mechanism. Nevertheless, we could imagine processes working per order and per degree or other logics that we would need to design.

Acknowledgments

This research has been supported by the EUR ArTeC financed by the French National Agency for Research ANR through the PIA ANR-17-EURE-0008. The author would like to thank Atau Tanaka for his advice and proof-reading. The author would like to thank also the *Maison des Sciences de l'Homme Paris Nord* for hosting this work and specially Anne Sèdes for her advice.

[7] like granulation, wider and rotation of the sound field.

5. REFERENCES

[1] J. Garcia, T. Carpentier, and J. Bresson, "Interactive-compositional Authoring of Sound Spatialization," *Journal of New Music Research*, vol. 46, no. 1, pp. 74–86, nov 2016.

[2] D. Fober, J. Bresson, P. Couprie, and Y. Geslin, "Les nouveaux espaces de la notation musicale," in *Journées d'Informatique Musicale*. Montréal, Canada: Faculté de musique de l'Université de Montréal, 2015. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01160759

[3] J. Blaeurt, *Spatial Hearing: The Psychophysics of Human Sound Localization*. Cambridge, Mass: MIT Press, 1983.

[4] B. Moore, *An Introduction to the Psychology of Hearing*, 6th ed. Brill Academic Pub, 2013.

[5] J. Colafrancesco, "Spatialisation de sources auditives étendues : applications musicales avec la bibliothèque HOA," phdthesis, Université Paris 8 Vincennes Saint-Denis, 2015.

[6] T. Pihlajamäki, O. Santala, and V. Pulkki, "Synthesis of spatially extended virtual source with time-frequency decomposition of mono signals," *J. Audio Eng. Soc*, vol. 62, no. 7/8, pp. 467–484, aug 2014. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=17339

[7] D. Kim-Boyle, "Spectral and Granular Spatialization with Boids," in *ICMC*. Michigan Publishing, 2006, pp. 139–142.

[8] A. Bonardi, "Composer l'espace sonore," in *Revue Francophone d'Informatique Musicale*, no. 7-8 Culture du code, 2020. [Online]. Available: https://revues.mshparisnord.fr/rfim/index.php?id=624

[9] A. Sedes, *Espaces sonores actes de recherche*. Paris: éd. musicales transatlantiques, 2003.

[10] A. Sèdes, P. Guillot, and E. Paris, "The HOA library, review and prospects," in *International Computer Music Conference — Sound and Music Computing*, ser. Proceedings ICMC—SMC, Athènes, Greece, Sep. 2014, pp. 855 –860. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01196453

[11] A. Bonardi, "La librairie abclib : un ensemble de codes Faust rassemblant 20 ans de recherche, enseignement et création en musique mixte," in *Journées d'Informatique Musicale 2021*. Visioconférences, France: AFIM, Jul. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03313611

[12] M. Stroppa, *Accorder musicalement un espace réel et un espace inventé*. Paris: Presses universitaires de France, 2007.

[13] N. Barrett, "Spatio-musical composition strategies," *Organised Sound*, vol. 7, no. 3, p. 313–323, 2002.

[14] H. Vaggione, *L'espace composable. Sur quelques catégories opératoires dans la musique électroacoustique*. L'Harmattan, 1998.

[15] Y. Etienne, *De l'espace sonore*, Les presses du réel, Ed. Strasbourg, France Dijon, France Monts, Frances: Haute école des arts du Rhin, Les Presses du réel Présence graphique, 2014.

[16] M. F. Franz Zotter, *Ambisonics*. Springer International Publishing, May 2019. [Online]. Available: https://www.ebook.de/de/product/35683429/franz_zotter_matthias_frank_ambisonics.html

[17] A. Bonardi and P. Guillot, "Concevoir des traitements ambisoniques en 2D et en 3D : l'exemple de Pianotronics 2," in *Journées d'Informatique Musicale*, ser. Actes des Journées d'Informatique Musicale, vol. 2015. Montréal, Canada: Olivier Bélanger and Nicolas Bernier and Julie Delisle and Pierre Michaud and Robert Normandeau and Caroline Traube, May 2015, p. 8. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01199688

[18] P. Goutmann, "Traitement spatial du son par décorrélation des signaux en ambisonie d'ordre élevé," in *Journées d'Informatique Musicale 2021*. Visioconférences, France: AFIM, Jul. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03313616

[19] A. Sèdes, "Approche musicale de la decorrelation microtemporelle dans la bibliothèque HOA," in *Journées d'Informatique Musicale 2015*, université de Montréal, Ed. Montréal, Canada: Université de Montréal, May 2015. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01199014

[20] G. S. Kendall, "The decorrelation of audio signals and its impact on spatial imagery," *Computer Music Journal*, vol. 19, no. 4, p. 71, 1995.

[21] H. Vaggione, "Décorrélation microtemporelle, morphologies et figurations spatiales," in *Journées d'Informatique Musicale*, Marseille, France, May 2002. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02992872

6. APPENDIX

List of sound examples quoted in the paper with short descriptions and links to HAL repository. All audio examples are mono sound encoded and processed in 2D 3^{th} order, and decoded in binaural (48kHz, 24bits).

6.1 Sound 1

- source: loop of an audio excerpt (10sec.) from the composition *Compression-Dilatation* 6.6
- description: the sound is presented five times, first without decorrelation (10sec.), second with the factor increasing, third with the factor at 1, fourth with the factor decreasing in 10 seconds and finally the sound without decorrelation (10sec.) ;
- parameter: delay = 1000ms, functiontype = 0, factor goes from 0 to 1 in 10 seconds then return to 0 in 10 seconds ;
- `https://hal.archives-ouvertes.fr/hal-03637787`

6.2 Sound 2

- source: loop ofa clarinet playing A3 ordinario ;
- description: we hear the effect of the factor of ring modulation ;
- parameter: f0 = 10, factor goes from 0 to 1 in 10 seconds and after 5 seconds returns to 0 in 10 seconds after 5 seconds ;
- `https://hal.archives-ouvertes.fr/hal-03637789`

6.3 Sound 3

- source: looped sample of a doublebass doing a Bartok pizzicato ;
- description: we hear the insertion of static delay line when the factor increases. First one echo, then two,*etc.*, until we heard seven distinct echos (each harmonic - at the third order in 2D) ;
- parameter: delay = 1000ms, functiontype = 0, factor goes from 0 to 1 in 25 seconds ;
- `https://hal.archives-ouvertes.fr/hal-03637800`

6.4 Sound 4

- source: looped sample of a doublebass doing a Bartok pizzicato ;
- description: we hear the delay time of each harmonic increasing progressively when the factor increases ;
- parameter: delay = 1000ms, functiontype = 0, factor goes from 0 to 1 in 25 seconds ;
- `https://hal.archives-ouvertes.fr/hal-03637802`

6.5 Sound 5

- source: looped sample of a doublebass doing a Bartok pizzicato ;
- description: we hear the effect of the different delay distribution between the ambisonics component ;
- parameter : delay = 1000ms, factor = 1, fd = 0, functiontype goes from 0 to 22 ;
- `https://hal.archives-ouvertes.fr/hal-03637803`

6.6 Sound 6

- title: *Compression - Dilatation* ;
- description: this is a sketch of electroacoustic piece composed in the context of the Ph.D research of Goutmann ;
- `https://hal.archives-ouvertes.fr/hal-03642163`

MODULAR PHYSICAL MODELS IN A REAL-TIME INTERACTIVE APPLICATION

Silvin Willemsen, Titas Lasickas, and Stefania Serafin
Multisensory Experience Lab, CREATE,
Aalborg University Copenhagen, Denmark
sil@create.aau.dk

ABSTRACT

Through recent advances in processing power, physical modelling using finite-difference time-domain (FDTD) methods has gained popularity. Many different musical instrument models based on these methods exist, and nearly all are based on the same underlying systems and interactions between them. This paper presents an application where individual resonator modules, such as strings, bars, membranes and plates, can be connected in a modular fashion and interacted with in real time. Various excitations, including the bow, hammer and pluck, are implemented as well, allowing for expressive control and a wide sonic palette. Existing and non-existing model configurations can easily be implemented, modified and experimented with, as well as the parameters describing them.

1. INTRODUCTION

Any acoustic musical instrument can be considered as the combination of a resonator and an exciter component [1]. Examples of resonator-exciter combinations are the violin and the bow, and the guitar and the pick. Many resonators, such as those mentioned here, can be further subdivided into basic components, i.e., a set of individual strings, a bridge and a wooden body. As many instruments consist of the same basic resonators – with different geometries or made from different materials – one can imagine some application that can implement many musical instruments based on the same fundamental resonator components in a modular fashion. Using physical modelling to implement these resonators allows for accurate implementation of the resonators, as well as the interactions between them.

Modularity in physical modelling sound synthesis is by no means a new concept. The earliest example of a modular system for sound synthesis was due to Cadoz *et al.* [2], with the CORDIS system. CORDIS allows complex instruments to be created using simple mass-spring interactions. Later, modular systems based on mass-spring systems appeared in various publications [3–5]. Morrison and Adrien created Mosaic [6], a modular environment using modal synthesis [7], more recently used by van Walstijn *et al.* in [8] and the Sound Design Toolkit [9]. Rabenstein *et al.* presented block-based modular physical models using wave digital filters [10] and digital waveguides [11].

Finite-difference time-domain (FDTD) methods, first used in a musical context by Ruiz [12], Hiller and Ruiz [13, 14] and later by Chaigne [15], also lend themselves to modularity. In [16], Bilbao presents a modular environment where bars and plates are connected by nonlinear springs, and in [17] Bilbao *et al.* propose a modular environment including higher dimensional systems.

Due to recent increase in computational power, FDTD methods have gained popularity in real-time applications [18]. A real-time modular environment using strings and lumped objects is presented in [19], and Südholt *et al.* present a real-time implementation of connected strings and bars using the FAUST programming language in [20].

The main goal of the aforementioned literature on FDTD-based modular environments is to create arbitrary sound-generators based on physical equations of motion, rather than modelling existing instruments. Furthermore, those able to run in real time only include systems spatially distributed over a maximum of one dimension. Many instruments require higher-dimensional structures to more accurately reproduce their sound. As done in e.g., [21, 22], the body of stringed instruments can be simplified to a thin plate, which is a system distributed over two dimensions.

This work presents an interactive modular environment in a real-time application where FDTD implementations of strings, bars, membranes and plates can be connected and played by the user. Although it is still possible to create arbitrary configurations to build non-existing instruments, the focus of this contribution is to allow for the possibility of modelling existing instruments. The current work aims to generalise previous work done on musical instruments modelled using FDTD methods (see e.g. [21, 22]), such that these can all be easily implemented as 'presets' of a modular application. Furthermore, some novel additions to the formulations of the hammer and pluck excitations are presented, required for their real-time interactions with the various resonators. The ultimate goal of this work is to act as a sound engine for a virtual reality application, potentially allowing for a more natural interaction.

This paper is structured as follows: Section 2 describes the physical models used in this work and Section 3 describes how to implement these. Section 4 presents the real-time application, Section 5 presents results and discusses these, and concluding remarks appear in Section 6.

Copyright: © 2022 Silvin Willemsen et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. MODELS

Consider a state variable $q(\boldsymbol{x}, t)$ with time $t \geq 0$ (in s) and spatial coordinate $\boldsymbol{x} \in \mathcal{D}$, where the dimensions and definition of domain $\mathcal{D}$ depend on the system at hand. The dynamics of a system can then be written using the following general form:

$$\mathcal{L}q = 0, \tag{1}$$

where linear partial differential operator $\mathcal{L}$ describes the dynamics of a model in isolation.

2.1 1D Systems

A commonly used 1D model (see e.g. [19, 21]) is the damped stiff string (or string for short). With reference to Eq. (1), consider a string of length L (in m), its transverse displacement described by state variable $q = u(\chi, t)$ (in m), and spatial coordinate χ is defined over domain $\mathcal{D} = [0, L]$. Furthermore, $\mathcal{L} = \mathcal{L}^{(1)}$ is defined as [23]

$$\mathcal{L}^{(1)} = \rho A \partial_t^2 - T\partial_\chi^2 + EI\partial_\chi^4 + 2\sigma_0 \rho A \partial_t - 2\sigma_1 \rho A \partial_t \partial_\chi^2, \tag{2}$$

where ∂_t and ∂_χ denote partial differentiation with respect to time and space respectively. The model is parameterised by material density ρ (in kg/m^3), cross-sectional area $A = \pi r^2$ (in m^2), radius r (in m), tension T (in N), Young's modulus E (in Pa), area moment of inertia $I = \pi r^4/4$ (in m^4) and loss coefficients $\sigma_0 \geq 0$ (in s^{-1}) and $\sigma_1 \geq 0$ (in m^2/s). If $T = 0$, the model reduces to a damped bar and the (damped) 1D wave equation if $E = 0$. Note that a circular cross-section is assumed here. Finally, in this work, the boundary conditions are chosen to be simply supported as

$$u = \partial_\chi^2 u = 0, \quad \text{for} \quad \chi = 0, L. \tag{3}$$

2.2 2D Systems

Consider a rectangular stiff membrane (or membrane for short) with side lengths L_x and L_y (both in m). With reference to Eq. (1), its transverse displacement can be described by $q = w(x, y, t)$ (in m), which is defined for domain $\mathcal{D} = [0, L_x] \times [0, L_y]$, and $\mathcal{L} = \mathcal{L}^{(2)}$ is defined as [24]

$$\mathcal{L}^{(2)} = \rho H \partial_t^2 - T\Delta + D\Delta\Delta + 2\sigma_0 \rho H \partial_t - 2\sigma_1 \rho H \partial_t \Delta, \tag{4}$$

Here, $\Delta = \partial_x^2 + \partial_y^2$ is the Laplacian, and parameters are material density ρ (in kg/m^3), thickness H (in m), tension per unit length T (in N/m), stiffness coefficient $D = EH^3/12(1-\nu^2)$ (in kg $\cdot$ m$^2\cdot$s^{-2}), Young's modulus E (in Pa), dimensionless Poisson's ratio ν, and loss coefficients $\sigma_0 \geq 0$ (in s^{-1}) and $\sigma_1 \geq 0$ (in m^2/s). If $T = 0$, the model reduces to a thin plate, and if $D = 0$ it reduces to the (damped) 2D wave equation, which can be used to model a non-stiff membrane. For simplicity, boundary conditions are chosen to be clamped, such that

$$w = \mathbf{n} \cdot \nabla w = 0, \tag{5}$$

where ∇ denotes 'the gradient of', and $\mathbf{n}$ is a normal to the plate area at the boundary.

2.3 Connections

One can add connections between instances of the models presented above by extending the general form in Eq. (1). Also see Figure 1. Consider M models q_m indexed by $m \in \mathfrak{M}$, where $\mathfrak{M} = \{1, \ldots, M\}$ and C connections between them. A connection is indexed by $c \in \mathfrak{C}$ with $\mathfrak{C} = \{1, \ldots, C\}$, and is characterised by the indices of the models it connects – $r_c \in \mathfrak{M}$ and $s_c \in \mathfrak{M}$ – and the locations where these models are connected – $\boldsymbol{x}_{r,c} \in \mathcal{D}_{r_c}$ and $\boldsymbol{x}_{s,c} \in \mathcal{D}_{s_c}$. Here, domains $\mathcal{D}_{r_c}$ and $\mathcal{D}_{s_c}$ are the (spatial) domains that models q_{r_c} and q_{s_c} are defined for.

Model q_{r_c} will be placed 'below' q_{s_c} such that the connection force acts positively on the former and negatively on the latter. The general form in Eq. (1) can then be extended to include connections according to

$$\mathcal{L}_m q_m = \sum_{\substack{c \in \mathfrak{C} \\ r_c = m}} \delta(\boldsymbol{x}_m - \boldsymbol{x}_{r,c}) f_c - \sum_{\substack{c \in \mathfrak{C} \\ s_c = m}} \delta(\boldsymbol{x}_m - \boldsymbol{x}_{s,c}) f_c, \tag{6}$$

where $f_c = f_c(t)$ is the force (in N) of the c^{th} connection.

The simplest connection considered in this work is the rigid connection, which assumes that the connected systems have an identical displacement at their respective connection locations. Alternatively, as done in [16, 24], one can use a spring to connect two systems. A connection force due to a nonlinear damped spring is

$$f_\text{c} = K_{1,c}\eta_c + K_{3,c}\eta_c^3 + R_c \partial_t \eta_c, \tag{7}$$

with is a linear spring coefficient $K_{1,c} \geq 0$ (in N/m), nonlinear spring coefficient $K_{3,c}$ (in N/m^3), and damping coefficient $R_c \geq 0$ (in s^{-1}) of the c^{th} connection. If $K_{3,c} = 0$, Eq. (7) reduces to a linear spring. Furthermore, $\eta_c = \eta_c(t) = q_{s_c}(\boldsymbol{x}_{s,c}, t) - q_{r_c}(\boldsymbol{x}_{r,c}, t)$ is the relative displacement of the two systems at their respective connection locations (in m). Section 3.2 will elaborate on how to calculate the connection forces for all cases in discrete time.

To illustrate, consider two models ($M = 2$), a string and a membrane, such that $q_1 = u(\chi, t)$ and $q_2 = w(x, y, t)$, and a single connection between them ($C = 1$) at unspecified locations $\chi_\text{c} \in \mathcal{D}_{s_1}$ and $(x_\text{c}, y_\text{c}) \in \mathcal{D}_{r_1}$ respectively. See Figure 1. Placing the string above the membrane, we get that $s_1 = 1$ and $r_1 = 2$, i.e., the 'above' model of connection 1 has index 1, and the 'below' model of connection 1 has index 2. Substituting the partial differential operators for the string and membrane found in Eqs. (2) and (4) respectively, Eq. (6) becomes, for the string and the membrane respectively

$$\mathcal{L}^{(1)} u = -\delta(\chi - \chi_\text{c}) f_1, \tag{8a}$$

$$\mathcal{L}^{(2)} w = \delta(x - x_\text{c}, y - y_\text{c}) f_1. \tag{8b}$$

One can apply a rigid connection to Eq. (8a) according to [24]:

$$u(\chi_\text{c}, t) = w(x_\text{c}, y_\text{c}, t). \tag{9}$$

If instead a spring connection is chosen,

$$f_1 = K_{1,1}\eta_1 + K_{3,1}\eta_1^3 + R_1 \partial_t \eta_1, \tag{10}$$

with $\eta_1 = \eta_1(t) = u(\chi_c, t) - w(x_c, y_c, t)$.

2.4 Excitations

In this work, as excitations will only be applied to 1D systems, the state variable of the stiff string, $u(\chi, t)$ will be used for the presentation of the various excitations. For the string, the general form in Eq. (1) can thus be extended to (ignoring connections for now)

$$\mathcal{L}^{(1)}u = e_\mathrm{e} f_\mathrm{e}, \tag{11}$$

where $e_\mathrm{e} = e_\mathrm{e}(\chi)$ is an excitation distribution and $f_\mathrm{e} = f_\mathrm{e}(t)$ is the externally supplied excitation force (in N).

2.4.1 The Bow

As done in previous work, see e.g. [21], one can include a bowing interaction by introducing a static friction model. With reference to Eq. (11), the excitation force can be defined using the following friction model [24]

$$f_\mathrm{e} = -f_\mathrm{B}\sqrt{2a_\mathrm{B}}v_\mathrm{rel}e^{-a_\mathrm{B}v_\mathrm{rel}^2+1/2}, \tag{12}$$

with externally supplied bow force $f_\mathrm{B} = f_\mathrm{B}(t)$ (in N), dimensionless free parameter a_B and

$$v_\mathrm{rel} = \partial_t u(\chi_\mathrm{B}, t) - v_\mathrm{B} \tag{13}$$

is the relative velocity (in m/s) between the string at externally supplied bowing location $\chi_\mathrm{B} = \chi_\mathrm{B}(t)$ (in m) and the externally supplied bow velocity $v_\mathrm{B} = v_\mathrm{B}(t)$ (in m/s). Furthermore, the excitation distribution is set to be a single point along the string:

$$e_\mathrm{e} = \delta(\chi - \chi_\mathrm{B}). \tag{14}$$

2.4.2 Hammer

Another way of exciting a system is to use a hammer, which can be modelled as a simple mass-spring-damper system with a few proposed additions to allow for real-time interaction. In this work, state variable $z = z(t)$ is used to describe the displacement of the hammer from its equilibrium position (in m). The interaction between the hammer and the string is then modelled as a collision, using the following collision potential [25]:

$$\phi(\eta_\mathrm{e}) = \frac{K_\mathrm{e}}{\alpha_\mathrm{e}+1}[\eta_\mathrm{e}]_+^{\alpha_\mathrm{e}+1}, \tag{15}$$

with collision stiffness $K_\mathrm{e} \geq 0$ (in N/m$^{\alpha_\mathrm{e}}$) and nonlinear collision coefficient $\alpha_\mathrm{e} \geq 1$. Furthermore, $[\eta_\mathrm{e}]_+ = 0.5(\eta_\mathrm{e} + |\eta_\mathrm{e}|)$ describes the 'positive part of η_e' and

$$\eta_\mathrm{e} = \eta_\mathrm{e}(t) = \theta\big(u(\chi_\mathrm{e}, t) - z(t)\big) \tag{16}$$

is the relative displacement between the string at collision location χ_e (in m) and the hammer (in m). Here, we propose $\theta = \tau$ if the string should be excited from above, and $\theta = -\tau$ if it should be excited from below, where $\tau = 1$ if the hammer interaction is triggered by the user and $\tau = 0$ if not (see Section 3.3.2).

Using a change of variables based on energy quadratisation proposed by Lopes *et al.* in [26] and used for collisions in a musical acoustics context in [27], one can rewrite the collision potential as

$$f_\mathrm{e} = -\theta\psi\psi', \tag{17}$$

where $\psi = \psi(\eta) = \sqrt{2\phi}$, and using dots to denote a temporal derivative, $\psi' = \dot{\psi}/\dot{\eta}$. This change of variables ultimately allows for an explicit implementation of the nonlinear collision.

The dynamics of the hammer, including the collision with the string, can be described by the following PDE:

$$M_\mathrm{z}\partial_t^2 z = -K_\mathrm{z}\Big(z - (1-\tau)z_\mathrm{off}\Big) - R_\mathrm{z}\partial_t z + \theta\psi\psi', \tag{18}$$

with mass M_z (in kg), spring constant K_z (in N/m), loss coefficient R_z (in kg/s) and $z_\mathrm{off} = z_\mathrm{off}(t)$ is an externally supplied offset (in m) (also see [22]). The offset is used to keep the mass away from the equilibrium (and thus the system it excites) when controlling the application without wanting to excite the system. If the hammer excitation is triggered, and thus $\tau = 1$, the offset will no longer have an effect on the mass. The spring force then pulls the hammer towards the system, colliding with it in the process. After the collision, $\tau = 0$, and the offset will affect the mass again to avoid continuous collision between the hammer and the system.

Finally, with reference to Eq. (11), the excitation distribution is set to be a raised cosine with centre location χ_e and excitation width e_w (in m):

$$e_\mathrm{e}(\chi) = \begin{cases} \frac{1-\cos\left(\frac{2\pi(\chi-\chi_\mathrm{e})}{e_\mathrm{w}}+\pi\right)}{2}, & \text{if } \chi_\mathrm{e} - \frac{e_\mathrm{w}}{2} \leq \chi \leq \chi_\mathrm{e} + \frac{e_\mathrm{w}}{2}, \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Note that χ_e and e_w must be chosen such that $\chi_\mathrm{e} - \frac{e_\mathrm{w}}{2} \in \mathcal{D}$ and $\chi_\mathrm{e} + \frac{e_\mathrm{w}}{2} \in \mathcal{D}$, where $\mathcal{D}$ is the domain of the excited system.

2.4.3 Pluck

The pluck is modelled nearly the same way as the hammer. The main difference is that $\tau = 1$ until the collision force is larger than a certain value, after which it will be set to 0. This will be elaborated on in Section 3.3. See Figure 1 for an example of the plucking interaction with a (connected) string.

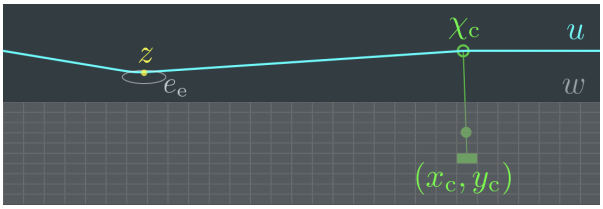


Figure 1. Snapshot of the application including a string and a thin plate with a rigid connection between them. The string is excited using a pluck.

3. DISCRETE TIME

In order to implement the models described in Section 2 using FDTD methods, a spatio-temporal grid needs to be

defined [24]. For all models, time is discretised to $t = nk$ with time index $n = 0, 1, 2\ldots$ and time step $k = 1/f_\mathrm{s}$ (in s) where f_s is the sample rate (in Hz). For the 1D systems, space is subdivided into N equal intervals of length h_s (in m) according to $\chi = ph$ with spatial index $p \in \{0, \ldots, N\}$. In the case of the 2D systems, the spatial coordinate is discretised as $(x, y) = (lh, mh)$ where spatial indices $l \in \{0, \ldots, N_x\}$ and $m \in \{0, \ldots, N_y\}$. Here, N_x and N_y are the number of intervals in the x and y direction respectively. Notice that the same value for grid spacing h is used for both the x and y directions.

Using these definitions, the general state variable $q(\boldsymbol{x}, t)$ can be approximated to grid function $q^n_{\boldsymbol{l}}$, where for the 1D systems $\boldsymbol{l} = p$ yielding grid function u^n_p and for the 2D systems, $\boldsymbol{l} = (l, m)$ yielding grid function $z^n_{(l,m)}$.

One of the main concerns when working with FDTD methods is the stability of the scheme. The stability condition for the discrete models can be described in terms of the grid spacing h as [18]

$$h \geq \sqrt{\beta\left(c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}\right)} \tag{20}$$

where for the string $c^2 = T/\rho A$, $\kappa^2 = EI/\rho A$ and $\beta = 0.5$, and for the membrane $c^2 = T/\rho H$, $\kappa^2 = D/\rho H$ and $\beta = 1$. The number of intervals between grid points can then be calculated as $N = \lfloor L/h \rfloor$ for 1D systems and $N_x = \lfloor L_x/h \rfloor$ and $N_y = \lfloor L_y/h \rfloor$ for 2D systems, where $\lfloor\cdot\rfloor$ denotes the flooring operation. Including the boundaries, discrete systems will have $N + 1$ and $(N_x + 1)(N_y + 1)$ grid points for the 1D and 2D case respectively.

3.1 FDTD schemes

The general form in Eq. (1) can be discretised to the following FDTD scheme:

$$\ell q^n_{\boldsymbol{l}} = 0 \tag{21}$$

where ℓ is the discretised version of linear partial differential operator $\mathcal{L}$. The discrete-time definitions of Eqs. (2) and (4) will not be given here for brevity, but can be found in the literature (e.g. [24], [18]). However, for any explicit FDTD scheme (which are used in this work), one can expand Eq. (21) to yield an update equation of the form

$$aq^{n+1}_{\boldsymbol{l}} = bq^n_{\boldsymbol{l}} + cq^{n-1}_{\boldsymbol{l}} \tag{22}$$

where a, b and c depend on the system at hand. In the following, we assume that $a = (1 + \sigma_0 k)$ for any discretised system.

3.2 Connections

To apply the effect of connections to FDTD schemes, interpolation and spreading operators must be introduced. As in Section 2.3, consider M models where the grid function of the m^{th} model is $q^n_{m,\boldsymbol{l}_m}$. One can define a (zeroth-order) interpolation operator as

$$I_{m,\boldsymbol{l}_m}(\boldsymbol{x}_m) = \begin{cases} 1, & \text{if } \boldsymbol{l}_m = \lfloor \boldsymbol{x}_m/h_m \rfloor, \\ 0, & \text{otherwise,} \end{cases} \tag{23}$$

which can be applied to grid function $q^n_{m,\boldsymbol{l}_m}$ to to obtain the state of one grid point of the system. A spreading operator can then be defined as

$$J_{m,\boldsymbol{l}_m}(\boldsymbol{x}_m) = \frac{1}{(h_m)^\varepsilon} I_{m,\boldsymbol{l}_m}(\boldsymbol{x}_m), \tag{24}$$

where ε is the number of spatial dimensions the system is defined over ($\varepsilon = 1$ for 1D systems, $\varepsilon = 2$ for 2D systems). Equation (24) can then be used to localise a (connection) force onto a specific location along a system. The general form in Eq. (6) can be discretised to

$$\ell_m q^n_{m,\boldsymbol{l}_m} = \sum_{\substack{c\in\mathfrak{C} \\ r_c = m}} J_{m,\boldsymbol{l}_m}(\boldsymbol{x}_{r,c}) f^n_c - \sum_{\substack{c\in\mathfrak{C} \\ s_c = m}} J_{m,\boldsymbol{l}_m}(\boldsymbol{x}_{s,c}) f^n_c, \tag{25}$$

Assuming that none of the connections overlap one can solve for each force f^n_c individually.

One can express the relative distance between two models (q_{s_c} and q_{r_c}) connected by connection c as

$$\eta^n_c = I_{s_c,\boldsymbol{l}_{s_c}}(\boldsymbol{x}_{s,c}) q^n_{s_c,\boldsymbol{l}_{s_c}} - I_{r_c,\boldsymbol{l}_{r_c}}(\boldsymbol{x}_{r,c}) q^n_{r_c,\boldsymbol{l}_{r_c}}. \tag{26}$$

If a rigid connection is chosen, the force of connection c can be solved for according to [24]

$$f^n_c = \frac{\eta^\star_c}{\frac{k^2}{a_{r_c}\mathcal{M}_{r_c}} + \frac{k^2}{a_{s_c}\mathcal{M}_{s_c}}}, \tag{27}$$

where $\mathcal{M}_m$ is the effective mass of a single grid point (in kg) of model q_m, i.e., $\mathcal{M} = \rho A h$ for 1D systems, and $\mathcal{M} = \rho H h^2$ for 2D systems and $a_m = (1 + \sigma_{0,m} k)$. Furthermore,

$$q^\star_{m,\boldsymbol{l}_m} = \frac{b_m q^n_{m,\boldsymbol{l}_m} + c_m q^{n-1}_{m,\boldsymbol{l}_m}}{a_m} \tag{28}$$

is the state of model q_m at the next time step in isolation, i.e., without the connection forces (obtained by rewriting Eq. (22)), and can be appropriately used in Eq. (26) to obtain $\eta^\star_c$.

Introducing the centred difference and centred averaging operator

$$\delta_t . \eta^n = \frac{1}{2k}\left(\eta^{n+1} - \eta^{n-1}\right) \approxeq \eta, \tag{29}$$

$$\mu_t . \eta^n = \frac{1}{2}\left(\eta^{n+1} + \eta^{n-1}\right) \approxeq \eta, \tag{30}$$

the spring in Eq. (7) can be discretised to

$$f^n_\mathrm{c} = K_{1,\mathrm{c}} \mu_t . \eta^n_\mathrm{c} + K_{3,\mathrm{c}} (\eta^n_\mathrm{c})^2 \mu_t . \eta^n_\mathrm{c} + R_\mathrm{c} \delta_t . \eta^n_\mathrm{c}, \tag{31}$$

which can be shown to be inherently stable [16, 24]. The connection force can then be solved for according to [24]

$$f^n_c = \frac{\eta^\star_c + \frac{\gamma_{c,-}}{\gamma_{c,+}} \eta^{n-1}_c}{\frac{1}{\gamma_{c,+}} + \frac{k^2}{a_{r_c}\mathcal{M}_{r_c}} + \frac{k^2}{a_{s_c}\mathcal{M}_{s_c}}}, \tag{32}$$

where

$$\gamma_{c,\pm} = \frac{K_{1,c}}{2} + \frac{K_{3,c}(\eta^n_c)^2}{2} \pm \frac{R_c}{2k}. \tag{33}$$

Notice that the rigid connection force in Eq. (27) is a special case of Eq. (32) where $\eta^{n-1}_c = 0$ and $\gamma_{c,+} \to \infty$. See [18, Ch. 11] for a derivation and more details.

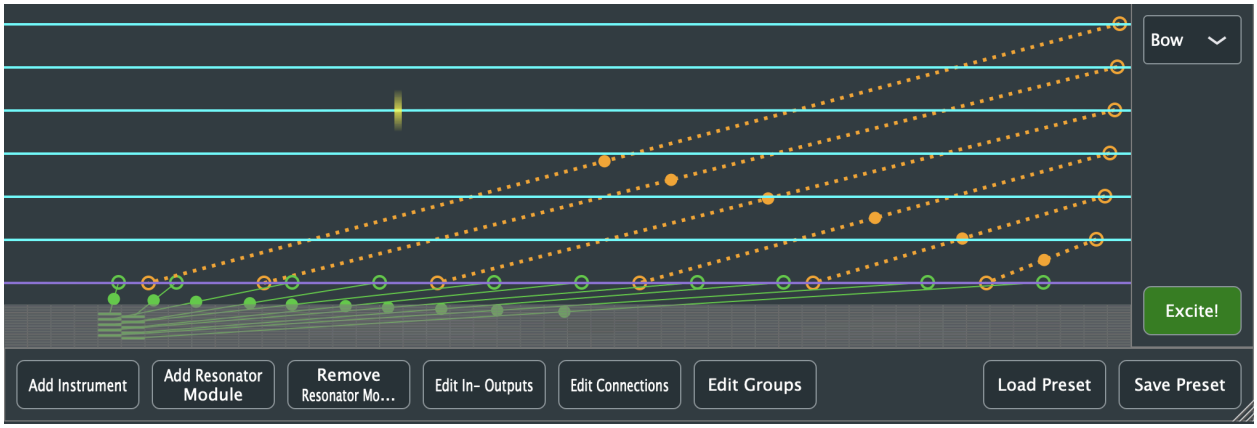


Figure 2. The graphical user interface (GUI) of the application presented in Section 4. Here, the guitar preset is loaded with 6 strings, 1 bar, 1 thin plate and several linear (orange) and rigid (green) connections between them, excited using a bow.

3.3 Excitations

3.3.1 The Bow

The discretisation of the friction model in Eq. (12) as well as its implementation using an iterative solver – such as Newton-Raphson – is well-covered in the literature (see e.g. [18, Ch. 8]). The spatial Dirac delta function in Eq. (14) is discretised using cubic spreading operator $J_{p,3}(\chi_\mathrm{i})$, rather than Eq. (24), for more accurate control. Its definition is also excluded here for brevity, but can be found in the literature [24, Sec. 5.2.4].

3.3.2 Hammer and Pluck

The discrete-time definition of Eq. (18) will not be given here. The discretisation of a mass-spring-damper system can be found in e.g. [16, 22], and a derivation of the collision between the mass-spring and a 1D system using the non-iterative collision method used here (from [27]) is given in [18, Sec. 10.2]. Changes in θ and τ (introduced in this work) are considered instantaneous and do not affect the derivations.

For the hammer, τ will be set to 0 if the user triggers this with the mouse (see Section 4.2.7). For the pluck, τ will be set to 0 if $|f_\mathrm{e}/h| > \varphi$, where φ is a threshold, which simulates a plucking interaction. Note that the force is scaled by the grid spacing of the respective resonator module so that the plucking interaction will be similar for strings with different parameters (and thus different values of h).

4. REAL-TIME APPLICATION

A real-time application implementing the models above has been created in C++ using the JUCE framework [1]. Figure 2 shows the graphical user interface of the application using the guitar preset for illustration. The application is controlled using the mouse, and is divided into three main parts: the control panel (bottom), the excitation panel (right), and the instrument area. The latter is vertically and equally subdivided over the amount of instruments in the application, which in turn are subdivided vertically and equally into the amount of resonators they contain. The instrument area has a refresh rate of 15 Hz and visualises the states of the various resonator and exciter modules.

After describing the system architecture, this section will go into detail of the functionality of the application. An extensive demo of the application, going through all of the functionality described in this section, can be found via [28].

4.1 System Architecture

Figure 3 illustrates the architecture of the audio-generating part of the application. The application can contain several independent instruments, each of which can contain several resonators. Furthermore, instruments contain information about the connections between various resonators. Every 1D resonator has three exciter modules, one for each of the excitations described in this paper. Only one of the exciter modules is activated at a time, controlled by the excitation panel (see Section 4.2.7).

4.2 Functionality

This section goes through functionality of the application following the order of the various buttons shown in the control panel (bottom area in Figure 2).

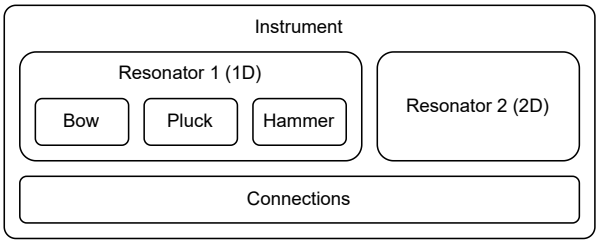


Figure 3. The system architecture. An instrument contains resonators as well as information about the connections between them. 1D resonators contain three exciter modules.

[1] https://juce.com

4.2.1 Instruments

The *Add Instrument* button adds an (additional) instrument to the application. Clicking on an instrument makes it the 'currently active instrument' to which resonator modules can be added.

4.2.2 Resonator Modules

The available resonators are: the stiff string, bar, membrane, thin plate and stiff membrane and are implemented as in Section 3.1. The states of the 1D models are visualised as cyan (string) and purple (bar) lines and the states of 2D models as grey-scale squares (as done in e.g. [21]).

The *Add Resonator Module* button opens a window where a user can select a resonator type as well as its parameters. For the stiff string, one can choose an advanced (all parameters) and non-advanced (only the fundamental frequency and radius) list of parameters. For 2D models, an extra parameter 'maxPoints' is given which alters the grid spacing such that the number of moving grid points, i.e., those that are not fixed at the boundaries, does not surpass this. This is to not overload the CPU and keep the application running in real time (also see Section 5). A button *Add Module*, adds the resonator module to the currently selected instrument.

The *Remove Resonator Modules* button changes the buttons in the control panel to *Remove* and *Done*. Clicking on a resonator module adds a red overlay, and clicking the *Remove* button removes the resonator from the instrument.

It must be noted that if any button is pressed, the states of all resonator modules will be set to 0 to prevent audible artefacts when editing the settings.

4.2.3 Outputs

The output of any model can be obtained by listening to $q^n_{\boldsymbol{l}_\mathrm{o}}$ for an output location $\boldsymbol{l}_\mathrm{o}$. In the application it is possible to change the output locations by clicking the *Edit In-Outputs* button. If the button is clicked, the control panel will show instructions on the chosen option as well as a *Done* button. The user can now add output locations to the various resonators.

For 1D systems, outputs will show as downwards pointing arrows from the system state at the respective output locations. For 2D systems, rectangles around the output grid point are used. Left, right and stereo channels are shown in white, red and yellow respectively. Functionality to add inputs has been left for future work (see Section 6).

4.2.4 Connections

Connections between resonators (implemented as in Section 3.2) are visualised using coloured (dotted) lines (also see Figure 2). Rigid connections are shown in solid green, linear springs in dotted orange and non-linear springs in dotted magenta. A connection location on a 1D system is accentuated using a circle of the same colour, and the same is done for a 2D system using a rectangle. A solid circle along the connection line indicates the mass ratio between the grid points of the two components: $\mathcal{M}_{s_c}/\mathcal{M}_{r_c}$ (see Eq. (27)). The closer this is to one component, the heavier a grid point of that component is with respect to the other.

If the *Edit Connections* button is clicked, the control panel will change in the same way as for the *Edit In- Outputs* button. Now, the currently active connection will be indicated by a yellow 'halo' around the connection locations. If a user tries to overlap two connections, the currently active connection will be removed from the application.

4.2.5 Groups

If the *Edit Groups* button is pressed, the control panel changes in the same way as for the previous two buttons. The user can click on 1D resonator modules to add them to groups (see [28] for more details). Grouped models can be excited simultaneously (see Section 4.2.7).

4.2.6 Presets

Presets are saved in '.xml' files and have the structure shown in Figure 4. Each preset contains several elements (instruments, resonators, connections, etc.), each containing one or more attributes.

Pressing the *Load Preset* button causes a 'File Chooser' window to pop up, where the user can select a local '.xml' file containing a preset. The *Save Preset* button makes a window pop up with a text box into which the desired name of the current application configuration can be saved. If a file with that name already exists a warning window will pop up asking whether the existing file may be overwritten.

As different sample rates yield different values for h (see Eq. (20)) and thus the number of grid points for each resonator, the locations of the outputs and connections are saved as ratios of the total number of grid points of the respective resonator. This allows for the same relative locations of outputs and connections for one preset between different sample rates.

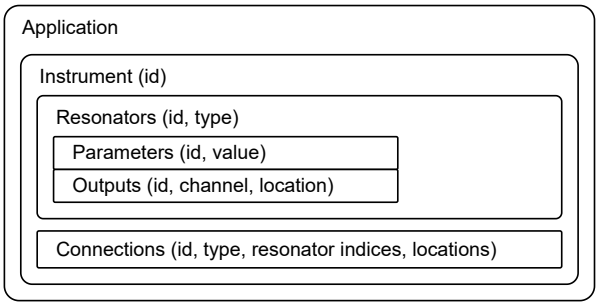


Figure 4. Structure of the '.xml' files containing presets. Attributes for several elements are given in parentheses.

4.2.7 Excitations

One can interact with the various 1D resonator modules in the instrument area. The excitation panel (right area in Figure 2) contains a dropdown menu with the various excitations: *Bow*, *Hammer* and *Pluck*. Selecting one of these activates the corresponding exciter module in all 1D resonator modules. Furthermore, an *Excite!* button toggles the currently active exciter module for all 1D resonator modules. Now a user can hover the mouse over the resonator modules in the instrument area (or click for the

Mouse action	Bow	Hammer & Pluck
x-position	χ_{B}	χ_{e}
y-position	-	z_{off}
Scroll-wheel	$-0.2 \leq v_{\mathrm{B}} \leq 0.2$	$h \leq e_{\mathrm{w}} \leq 10h$
Click	-	τ (hammer only)

Table 1. Mouse interactions related to the various excitations described in Section 4.2.7.

Name	Symbol (unit)	Value
Bow		
Free parameter	a_{B} (-)	100
Bow force	f_{B} (N)	$40 \cdot \rho A$
Hammer / pluck		
Mass	M_{z} (kg)	0.01
Spring coefficient	K_{z} (N/m)	1000
Loss coefficient	R_{z} (kg/s)	1
Collision stiffness	K_{e} (N/m$_{\mathrm{e}}^{\alpha}$)	10^6
Nonlin. coll. coeff.	α_{e} (-)	1.3
Pluck force threshold	φ (N/m)	500
Connections		
Linear spring coeff.	K_1 (N/m)	10^8
Nonlin. spring coeff.	K_3 (N/m^3)	10^{10}
Damping coeff	R (s^{-1})	0.01

Table 2. List of fixed parameter values (see Section 4.3).

hammer) to excite them. If the excitation is deactivated, all resonators (including 2D ones) can be excited using a raised cosine (although this is only used for testing purposes). Table 1 shows how the mouse is related to various excitation parameters.

The bow is visualised with a yellow rectangle with a moving gradient, the speed and direction of which depends on the value of the bow velocity (v_{B} in Eq. (13)). The mass used for the hammer and pluck (z in Eq. (18)), is visualised using a yellow circle and follows the mouse / cursor. The excitation distribution (e_{e} in Eq. (19)) is visualised using a grey ellipse around the cursor; its width is determined by the value of e_{e}. For the hammer, a mouse click sets $\tau = 0$ in Eq. (18).

4.3 Fixed parameters

As described in Section 4.2.2, the parameters of the resonator modules can easily be altered by the user when adding one to the application. The parameters found in Table 2, however, can not be altered by the user, and have been set to result in pleasing sounds for many different model configurations.

5. RESULTS AND DISCUSSION

Table 3 shows the CPU usage of the application with various settings and graphics turned off and on. Tests were carried out on a MacBook Pro with a 2,3 GHz Intel i9 processor. Results show that for many different configurations, the application is still able to run in real time (usage $< 100\%$).

Strings ($N = 118$)				
No. of strings	1	5	10	20
No graphics	1.7	7.4	12.3	26.8
Graphics	8.3	14.0	19.4	35.0
Plate				
Moving points	100	400	1600	2500
No graphics	4.8	15.4	47.0	63.3
Graphics	12.0	23.9	59.2	77.5
Two connected strings ($N = 118$, $C = 117$)				
Type	None	Rigid	Linear	Nonlinear
No graphics	3.8	8.3	13.1	13.3
Graphics	12.7	28.4	42.3	43.1

Table 3. CPU usage (in %) of the application with various configurations, and graphics turned on and off.

One can observe that the CPU increases with the number of grid points that need to be calculated per sample, which is an expected result. As more calculations need to be performed per sample for a 2D system the CPU usage is higher per grid point than for a 1D system. For two strings with all their moving grid points connected ($N = 118$, $C = 117$), the CPU usage increases significantly when compared to unconnected strings, and more so for non-rigid connections. Moreover, the graphics increase the CPU usage substantially, especially when connections need to be drawn. As the speed of the graphics thread has not been the focus during the development of the application, this could be optimised in the future.

To ensure that the application runs in real time, several things need to be considered. One example is the 'maxPoints' variable, limiting the number of grid points for 2D systems mentioned in Section 4.2.2. Using fewer grid points reduces the simulation quality and the frequency bandwidth of the model at hand [24]. Furthermore, it has been chosen to only use explicit models, which, although more computationally cheap, cause detuning of modes in models with high stiffness values such as bars and plates [24]. These effects, however, have been found to be less important for stringed instrument models, due to the large amount of damping of bars and plates and the inharmonic nature of models with high stiffness.

6. CONCLUSION

This paper presented an application implementing various resonator modules using FDTD schemes which can be connected in a modular fashion to create (non)existing musical instruments. The instruments can be played in real time using bowing, hammering and plucking excitations.

Future work could include to embed this contribution in a virtual reality application where users will be able to build and play instruments in a virtual environment.

Acknowledgments

This work has been funded in part by the European Art-Science-Technology Network for Digital Creativity (EASTN-DC), project number 883023.

7. REFERENCES

[1] G. Borin, G. De Poli, and A. Sarti, "A modular approach to excitator-resonator interaction in physical models syntheses," *Proc. Int. Comp. Music Conf.*, 1989.

[2] C. Cadoz, A. Luciani, and J.-L. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms: the CORDIS system," *Comp. Music J.*, vol. 8, no. 3, pp. 60–73, 1983.

[3] N. Castagné and C. Cadoz, "GENESIS: A friendly musician-oriented environment for mass-interaction physical modeling," in *Proc. Int. Comp. Music Conf.*, 2002, pp. 330–337.

[4] E. Berdahl and J. Smith, "An introduction to the synth-a-modeler compiler: Modular and opensource sound synthesis using physical models," in *Proc. 10th Int. Linux Audio Conf. (LAC-12)*, 2012, pp. 223–228.

[5] J. Leonard and J. Villeneuve, "MI-GEN~: An efficient and accessible mass interaction sound synthesis toolbox," in *Proc. 13th Int. Conf. Sound Music Comp. (SMC)*, 2019.

[6] J. D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Comp. Music J.*, vol. 17, no. 1, pp. 45–56, 1993.

[7] J.-M. Adrien, "The missing link: Modal synthesis," in *Representations of Musical Signals*, G. De Poli, A. Picalli, and C. Roads, Eds. MIT Press, 1991, pp. 269–298.

[8] M. van Walstijn and S. Mehes, "An explorative string-bridge-plate model with tunable parameters," in *Proc. 20th Int. Conf. on Digital Audio Effects (DAFx)*, 2017.

[9] S. Baldan, S. Delle Monache, and D. Rocchesso, "The sound design toolkit," *SoftwareX*, vol. 6, pp. 255–260, 2017.

[10] R. Rabenstein, S. Petrausch, A. Sarti, G. D. Sanctis, C. Erkut, and M. Karjalainen, "Block-based physical modeling for digital sound synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 42–54, 2007.

[11] J. O. Smith, "Physical modeling using digital waveguides," *Comp. Music J.*, vol. 16, no. 4, pp. 74–91, 1992.

[12] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," Master's thesis, University of Illinois, 1969.

[13] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *J. Acoust. Soc. Am.*, vol. 19, no. 6, pp. 462–470, 1971.

[14] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *J. Acoust. Soc. Am.*, vol. 19, no. 7, pp. 542–550, 1971.

[15] A. Chaigne, "On the use of finite differences for musical synthesis. Application to plucked stringed instruments," *Journal d'Acoustique*, vol. 5, no. 2, pp. 181–211, 1992.

[16] S. Bilbao, "A modular percussion synthesis environment," in *Proc. 12th Int. Conf. on Digital Audio Effects (DAFx)*, 2009.

[17] S. Bilbao, A. Torin, P. Graham, J. Perry, and G. Delap, "Modular physical modeling synthesis environments on GPU," in *Proc. ICMC|SMC|2014*, 2014, pp. 1396–1403.

[18] S. Willemsen, "The emulated ensemble: Real-time simulation of musical instruments using finite-difference time-domain methods," Ph.D. dissertation, Aalborg University Copenhagen, Jul. 2021.

[19] S. Bilbao, M. Ducceschi, and C. Webb, "Large-scale real-time modular physical modeling sound synthesis," in *Proc. 22th Int. Conf. on Digital Audio Effects (DAFx)*, 2019.

[20] D. Südholt, R. Russo, and S. Serafin, "A faust implementation of coupled finite difference schemes," in *Proc. 2nd Nordic Sound and Music Comp. (Nordic-SMC) Conf.*, 2021.

[21] S. Willemsen, N. Andersson, S. Serafin, and S. Bilbao, "Real-time control of large-scale modular physical models using the sensel morph," *Proc. 16th Int. Conf. Sound Music Comp. (SMC)*, pp. 275–280, 2019.

[22] S. Willemsen, S. Serafin, S. Bilbao, and M. Ducceschi, "Real-time implementation of a physical model of the tromba marina," in *Proc. 17th Int. Conf. Sound Music Comp. (SMC)*, 2020, pp. 161–168.

[23] J. Bensa, S. Bilbao, R. Kronland-Martinet, and J. O. Smith, "The simulation of piano string vibration: From physical models to finite difference schemes and digital waveguides," *J. Acoust. Soc. Am.*, vol. 114, no. 2, pp. 1095–1107, 2003.

[24] S. Bilbao, *Numerical Sound Synthesis*. John Wiley & Sons, 2009.

[25] H. Hertz, "Über die berührung fester elastischer körper," *Journal für die Reine und Angewandte Mathematik*, vol. 92, pp. 156–171, 1881.

[26] N. Lopes, T. Hèlie, and A. Falaize, "Explicit second-order accurate method for the passive guaranteed simulation of port-hamiltonian systems," in *Proc. 5th IFAC*, 2015, pp. 223–228.

[27] M. Ducceschi, S. Bilbao, S. Willemsen, and S. Serafin, "Linearly-implicit schemes for collisions in musical acoustics based on energy quadratisation," *J. Acoust. Soc. Am.*, vol. 149, pp. 3502–3516, 2021.

[28] S. Willemsen, "Modular instrument demo," 2021. [Online]. Available: https://youtu.be/o6I2DlFIbsc

RUBBING A PHYSICS BASED SYNTHESIS MODEL: FROM MOUSE CONTROL TO FRICTIONAL HAPTIC FEEDBACK

Marius George Onofrei
Dept. of Design & Media Technology
Aalborg University
Copenhagen, Denmark
monofr11@student.aau.dk

Federico Fontana
Human Computer Interaction Lab
University of Udine
Udine, Italy
federico.fontana@uniud.it

Stefania Serafin
Multisensory Experience Lab
Aalborg University
Copenhagen, Denmark
sts@create.aau.dk

ABSTRACT

This paper investigates three kinds of interactions for a friction based virtual music instrument. The sound synthesis model consists of a bank of mass-spring-dampers individually excited via rubbing. A nonlinear static friction model capable of reproducing the characteristic stick-slip phenomenon observed in frictional interaction is employed, allowing for dynamic variation of the sliding friction.

The different controls developed allow for gradually increasing the interplay between performer and instrument. The key excitation parameters, e.g., the rubbing velocity and the rubbing normal force are controlled using three different interfaces: a standard mouse, a Sensel Morph, and a 3D Systems Touch X. The Sensel Morph is a touchpad with pressure sensitivity, allowing for a natural exertion of the normal force; the 3D Systems Touch X is a haptic device that renders both resistance to the applied normal force, as well as the stick-slip motion resulting from the friction interaction.

A preliminary user study aiming to compare the experience of performing with the different interfaces was carried out. The results indicate that the haptic feedback provides a more intuitive and enjoyable experience. However, extra features do not necessarily improve the user interaction, as the results suggest a preference for the mouse over the Sensel.

1. INTRODUCTION

Virtual musical instruments can be defined as computer programs that generate digital audio, typically for music. The interaction with such software applications is, in essence, an interaction with the computer itself. The use of the mouse and/or keyboard as control interfaces to such instruments is a straightforward choice. Other forms of controllers designed specifically for electronic music instruments (including audio applications) exist, with the most popular one being the MIDI keyboard, which makes use of a standardized serial protocol, which allows for communication between controllers and synthesizers.

Using such interfaces, however, can introduce a disconnection between the gestural actions and the sound produced by the virtual instrument. In fact, Cadoz proposes to preserve a consistent link between traditional instrumental gestures and their sound [1]. This disconnect, present in most virtual instruments, is particularly evident when considering physics based sound synthesis, as pointed out by O'Modhrain, who highlights the example of using a piano keyboard to control a physical model of a trumpet [2]. Nuances in the timber and dynamics of the sound being played are lost in a simple note on/note off system. Extra layers of controls, such as knobs controlling an ADSR envelope, or the mod wheel could be used to add these different nuances, but at the cost of extra complexity and with loss of an intuitive musical link between action and perception.

Controllers that more accurately mimic the excitation mechanisms of the instruments being simulated could provide a more satisfying performance experience. Virtual reality (VR) environments prove to be a fruitful ground for such explorations. As an example, Willemsen et al. made use of the haptic device Phantom Omni, to control a model of the tromba marina - a bowed string instrument [3]. Fontana et al. made use of the VR environment Unity 3D, together with the same haptic device to build a simulation of a plucked guitar [4].

A focus on intuitive musical gestures is also a critical part of the new interfaces for musical expression (NIME) community, where for example a very popular iPhone audio app, Smule's Ocarina, was first introduced [5]. Part of the appeal of this app, which simulates a wind instrument, is the use of breath-control as excitation mechanism, adding a layer of realism to the interaction.

One could argue that the realistic replication of all aspects contributing to the interaction with a real instrument is redundant. Replicating the excitation mechanism in the digital domain, however, allows to push the synthetic model to physical limits and even beyond. Consider as an example the work of Huynh who in [6] explores the effects on perception when exciting resonators in unnatural ways: "bowing plates" or "blowing strings". Even when aiming to model an instrument one-to-one, the virtual domain allows for unconventional interactions, as Onofrei et al. showed in the case of a friction drum model where the position

Copyright: © 2022 Marius George Onofrei et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

of the friction excitation on the drum membrane could be changed in real-time, something not possible with a real instrument counterpart [7].

This paper explores the question of how to control a physics-based instrument model in an intuitive way, with the aim of providing a satisfying user experience. A simple audio application consisting of a bank of mass-spring-damper elements individually excited by means of rubbing was built. This simple and somewhat abstract physical interaction was chosen as to investigate three control strategies which were then developed with the aim of progressively affording musical "rubbing" gestures, as well as providing the corresponding haptic feedback. A preliminary user study was then carried out evaluating the different setups. An important note is that the friction feedback used in our system results from the actual simulation of the friction interaction; the haptic friction force is the same as the force which excites the sound synthesis model. This is something that was not attempted in previous similar research, e.g. in both [3] and [8] the friction haptic feedback resulted from a different model than the one used for auditory feedback.

In Section 2 we describe the physical model behind the sound synthesis and the numerical method used to solve the resulting system of equations. Section 3 deals with the implementation of the model: it presents results of our off-line prototype with the aim of validating the nonlinear static friction model, then introduces the real-time application by describing its various features and controls. Finally the three user interfaces and their respective control strategies are presented. An evaluation of the resulting setups and its outcome is presented next in Section 4 and final concluding remarks are given in Section 5.

2. SOUND SYNTHESIS

The sound synthesis presented in this paper is based on the mass-spring-damper resonator model excited via frictional interaction. This resonator is the mechanical equivalent of a damped harmonic oscillator, which produces decaying sinusoids, and is of central importance to musical sound synthesis, both in abstract sound synthesis algorithms such as additive, subtractive or FM synthesis, as well as in physical modelling synthesis [9].

2.1 Physical Model

The model consists of a mass m [kg] connected to a rigid support through a parallel system of a linear spring, of spring constant k [N/m], and linear damper, of damping coefficient c [kg/s], as illustrated in Fig. 1. This mass is then excited via a rigid and weightless stick, which is pressed onto it with normal force N [N] and moved with a rubbing velocity v_r [m/s]. Its displacement relative to rest position at time t [s] is denoted $u(t)$ [m] and can be described by the following 2nd order ordinary differential equation (ODE):

$$m\partial_t^2 u + c\partial_t u + ku = -F_{fr}, \tag{1}$$

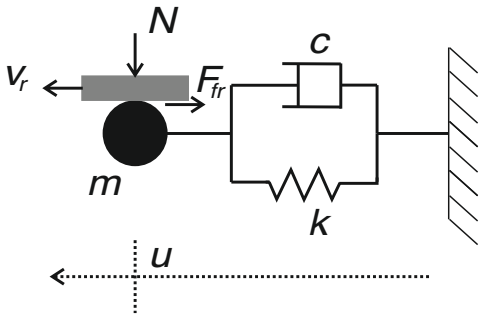


Figure 1: Mass-spring-damper system excited via friction.

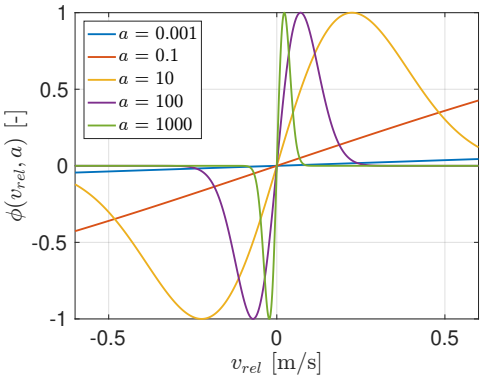


Figure 2: Shape of nonlinear friction characteristic given in Equation (3) for various values of a [s^2/m^2].

where F_{fr} [N] is the nonlinear frictional force resulting at the contact between the stick and the mass, modelled as:

$$F_{fr} = N\phi(v_{rel}, a). \tag{2}$$

This equation scales the applied normal force N with a nonlinear function $\phi(v_{rel}, a)$ which depends on the relative velocity between the stick and the mass, v_{rel} [m] and a friction parameter a [s^2/m^2] that gives the shape of the nonlinear function, and essentially controls how "sticky" the interaction is. The nonlinear function is defined in Equation (3), while Equation (4) gives the relative velocity as the difference between the time derivative of the mass's displacement and the rubbing velocity of the stick:

$$\phi(v_{rel}, a) = \sqrt{2a}v_{rel}e^{-av_{rel}^2+0.5}, \tag{3}$$

$$v_{rel} = \partial_t u - v_r. \tag{4}$$

Fig. 2 shows the nonlinear characteristic $\phi(v_{rel}, a)$ for various values of a. This friction model was introduced by Bilbao in [9] and it interpolates between an approximation of a viscous friction model (for small values of a) and a Stribeck model (for large values of a), where the discontinuity around zero relative velocity is smoothed out by the exponential function, making the friction characteristic continuous and differentiable, thus easier to compute numerically. Due to its stability and small number of parameters, which makes it more predictable and easy to work with, it has been fairly used for real-time audio simulations of bowed strings, e.g. in [10, 11].

For the case of the undamped system, i.e. the damping coefficient $c = 0$, it is well known that the natural resonance frequency of the system, f_0, is given by:

$$f_0 = \frac{1}{2\pi}\sqrt{k/m}. \tag{5}$$

Introducing damping will slightly skew this value by a factor determined by the critical damping factor, ζ, i.e., a ratio of the damping c over the amount of damping required to reach critical damping. The resonance frequency of the damped system, $f_{0,d}$ is given by:

$$\begin{aligned} f_{0,d} &= f_0\sqrt{1-\zeta^2}, \quad \text{with:} \\ \zeta &= \frac{c}{2m\sqrt{k/m}}. \end{aligned} \tag{6}$$

As can be deduced from the equations, the effect of damping on the resonance frequency becomes significant for values close to the critical damping, which is not the case for the current application.

With respect to the sound synthesis, the audio output is given by the displacement of the mass.

2.2 Solving the System

A numerical model can be figured out using the finite difference time domain (FDTD) method. Since the model is lumped, i.e. its parameters are not distributed in space, only the time domain needs to be discretized. The continuous time is sampled in discrete time steps indexed by $n \in \mathbb{N}$ as such: $t = nT$, where $T = 1/f_\text{S}$ [s] is the temporal step, f_S [Hz] being the sampling frequency.

With this discretization in place, the continuous function describing the displacement of the mass, $u(t)$, can be approximated by a grid function u^n, over the previously introduced temporal grid. Such discretizations are similarly carried out for the other functions: $F_{fr}(t) \approx F_{fr}^n$, $N(t) \approx N^n$, $v_r(t) \approx v_r^n$, $v_{rel}(t) \approx v_{rel}^n$ and $a(t) \approx a^n$. Moreover, the 1st and 2nd order time derivatives from Equation (1) can be approximated using the following operators:

$$\partial_t u \approx \delta_{t\cdot} u^n = \frac{1}{2T}\left(u^{n+1} - u^{n-1}\right), \tag{7a}$$

$$\partial_t^2 u \approx \delta_{tt} u^n = \frac{1}{T^2}\left(u^{n+1} - 2u^n + u^{n-1}\right). \tag{7b}$$

These approximations replace the continuous variables presented in the previous subsection, so that Equation (1) is discretized as:

$$m\delta_{tt}u^n + c\delta_{t\cdot}u^n + ku^n = -F_{fr}^n = -N^n\phi\left(v_{rel}^n, a^n\right). \tag{8}$$

From this, an update equation can be calculated for u^{n+1}:

$$u^{n+1} = u^n\frac{4m - 2kT^2}{2m + cT} - u^{n-1}\frac{2m - cT}{2m + cT} - F_{fr}^n\frac{2T^2}{2m + cT} \tag{9}$$

However, the resulting friction force at the current time step, F_{fr}^n, depends on the displacement of the mass at the following step, u^{n+1}. This dependence results from the insertion of Equation (7a) into Equation (4) and subsequently Equation (2). This issue can be addressed by making use of the following identity:

$$\delta_{tt}u^n = \frac{2}{T}\left(\delta_{t\cdot}u^n - \delta_{t-}u^n\right), \tag{10}$$

with $\delta_{t-}u^n = \frac{1}{T}\left(u^n - u^{n-1}\right)$ being a backward time-difference approximation to $\partial_t u$.

Substituting $\delta_{tt}u^n$ as such in Equation (8) and by making use of the definition of the relative velocity in the discrete domain:

$$v_{rel}^n = \delta_{t\cdot}u^n - v_r^n, \tag{11}$$

one can figure out the following equation:

$$\begin{aligned} f\left(v_{rel}^n\right) = \frac{N^n}{m}\phi\left(v_{rel}^n, a^n\right) + v_{rel}^n\left(\frac{c}{m} + \frac{2}{T}\right) + b^n = 0, \\ b^n = v_r^n\left(\frac{c}{m} + \frac{2}{T}\right) + u^n\left(\frac{k}{m} - \frac{2}{T^2}\right) + u^{n-1}\frac{2}{T^2}. \end{aligned} \tag{12}$$

This can be solved for v_{rel}^n using e.g. a Newton-Raphson iterative scheme, limited to a maximum of 99 iterations in order to avoid audio drop-outs. This value is then used to calculate F_{fr}^n, which allows for the solution to be propagated in time by computing u^{n+1} from Equation (9).

2.2.1 Stability

The stability of the finite difference scheme can be studied by means of an energy analysis of the discrete system, based on the work of [9] and [12]. Equation (8) is multiplied with $\delta_{t\cdot}u^n$, resulting in:

$$\begin{aligned} &m(\delta_{t\cdot}u^n)(\delta_{tt}u^n) + c(\delta_{t\cdot}u^n)^2 + \\ &k(\delta_{t\cdot}u^n)u^n = -(\delta_{t\cdot}u^n)N^n\phi\left(v_{rel}^n, a^n\right), \end{aligned} \tag{13}$$

Introducing $\delta_{t+}u^n = \frac{1}{T}\left(u^{n+1} - u^n\right)$ as the forward time-difference approximation to $\partial_t u$, the following identities hold:

$$(\delta_{t\cdot}u^n)(\delta_{tt}u^n) = \delta_{t+}\left(\frac{1}{2}(\delta_{t-}u^n)^2\right), \tag{14a}$$

$$(\delta_{t\cdot}u^n)u^n = \delta_{t+}\left(\frac{1}{2}(u^n u^n - 1)\right). \tag{14b}$$

Using these together with Equation (11), Equation (13) can be rewritten as:

$$\begin{aligned} &\delta_{t+}\left(\frac{m}{2}(\delta_{t-}u^n)^2\right) + \delta_{t+}\left(\frac{k}{2}(u^n u^n - 1)\right) = \\ &- c(\delta_{t\cdot}u^n)^2 - N^n v_{rel}^n\phi(v_{rel}^n) - N^n v_r^n\phi(v_{rel}^n). \end{aligned} \tag{15}$$

These terms have each an energetic interpretation. The first term in the left side of the equation is the discrete rate of change (δ_{t+}) of the kinetic energy, $\mathfrak{t}$, and the second is the rate of change in potential energy, $\mathfrak{v}$. Their sum gives the rate of change of the total energy, $\mathfrak{h}$.

$$\delta_{t+}\mathfrak{h}^n = \delta_{t+}\mathfrak{t}^n + \delta_{t+}\mathfrak{v}^n, \tag{16a}$$

$$\mathfrak{t}^n = \frac{m}{2}(\delta_{t-}u^n)^2, \tag{16b}$$

$$\mathfrak{v}^n = \frac{k}{2}(u^n u^{n-1}). \tag{16c}$$

Looking at the right side of Equation (15), the first term can be viewed as a loss in the system due to damping, $\mathfrak{q}$, while the next two terms represent the power dissipated and supplied by the excitation via the stick, $\mathfrak{p}_d$ and $\mathfrak{p}_s$:

$$\mathfrak{q}^n = -c(\delta_t.u^n)^2, \tag{17a}$$
$$\mathfrak{p}_d^n = -N^n v_{rel}^n \phi(v_{rel}^n), \tag{17b}$$
$$\mathfrak{p}_s^n = -N^n v_r^n \phi(v_{rel}^n). \tag{17c}$$

It is evident that $\mathfrak{q}^n \leq 0$, as the damping coefficient $c \geq 0$. Similarly, $\mathfrak{p}_d^n \leq 0$ since the normal rubbing force N^n cannot be negative and, following the definition of the friction characteristic $\phi(v_{rel}^n, a^n)$ given in Equation (3), sign(v_{rel}^n) = sign($\phi(v_{rel}^n, a^n)$). This leaves the supplied power, $\mathfrak{p}_s^n$, which has indeterminate sign. However, an upper bound can be found for it, as $|\phi(v_{rel}^n, a^n)| \leq 1$. It follows that $\mathfrak{p}_s^n \leq |N^n v_r^n|$, meaning that the rate of change of total energy, $\mathfrak{h}^n$, is bounded by the following inequality:

$$\delta_{t+}\mathfrak{h}^n \leq |N^n v_r^n|\,, \tag{18}$$

with the external excitation inputs N^n and v_r^n being finite and thus, assuring that the energy in the system will not explode.

Additionally, for guaranteeing stability of the numerical scheme, it must be ensured that the total numerical energy, $\mathfrak{h}^n$, cannot be negative. This is done by starting from Equation (16) and expanding the finite difference operators, then rearranging the equation in a quadratic form:

$$\left(\frac{2T^2}{m}\right)\mathfrak{h}^n = (u^n)^2 + (u^{n-1})^2 + 2\left(\frac{kT^2}{2m} - 1\right)u^n u^{n-1}. \tag{19}$$

Based on this formulation the following condition must hold in order for $\mathfrak{h}^n$ to be positive definite:

$$\left|\frac{kT^2}{2m} - 1\right| < 1, \tag{20}$$

from which the final stability condition on the temporal step T results:

$$T < 2\sqrt{\frac{m}{k}}. \tag{21}$$

3. IMPLEMENTATION

The aim was to implement a real-time polyphonic audio application, consisting of a bank of mass-spring-dampers tuned to various musical frequencies. Additionally it was desired to have dynamic control of the damping as well as the friction parameter a, which controls how "sticky" the interaction is.

3.1 Prototype Model Results

A prototype implementation of the sound synthesis model described in the previous section was first carried out in Matlab, whose offline setting allowed for a more in-depth analysis of the results. A particular focus was on whether the friction model produces results in line with the analytical results. Fig. 3 shows the displacement of a mass during a rubbing simulation of 0.1 seconds, meant to highlight the stick-slip behavior of the frictional interaction. The relative velocity between the stick and the mass periodically hovers around zero, meaning the two elements are stuck to each other until an abrupt slip, during which this velocity increases drastically. This results in a sawtooth waveform for the displacement of the mass and in similar sudden jumps for the resulting frictional force.

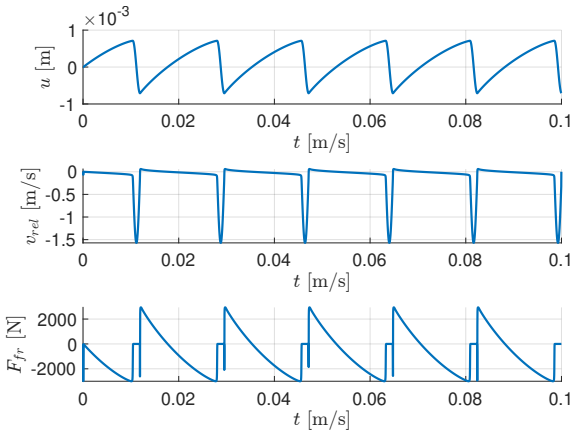


Figure 3: Results of a rubbing simulation with $m = 1$ [kg], $k = 4.2992$E6 [N/m], $c = 0$ [kg/s], $a = 100$ [s^2/m^2], $N = 3000$ [N], $v_r = 0.1$ [m/s], $f_S = 44100$ [Hz].

3.2 Real-Time Application

The real-time application was written in C++ using the JUCE framework. A demo video is available at [13] and Fig. 4 shows a snapshot of the application during use. Eleven mass elements are placed in the center of the application window, appearing as circles filled with different red tones. The darker the tone the lower the natural frequency. They are tuned starting from a resonance frequency of 55 Hz up to 330 Hz with a constant interval of 27.5 Hz. The simulation runs at a sampling frequency $f_S = 44100$ [Hz] and a corresponding temporal step $T = 0.0227$ [ms], ensuring numerical stability based on the condition given in Equation (21). The oscillation of these masses, i.e. the displacement obtained from the numerical simulation described in Section 2.2 is used as the audio output signal. This same displacement is updated in the application window at a rate of 24 frames per second, thus providing visual feedback to the user. Perhaps a better alternative for the sound output is the velocity of the masses, which is more proportional to the radiated sound of a physical instrument and also has more high-frequency detail. Two applications with the different sound output choice are available at [14]. These can be controlled using the mouse or a laptop's touch pad, as described in more detail in Section 3.3.

The model of the stick used to excite the masses is portrayed as a long rectangle with varying opacity. When the stick is not in contact with the masses, it is displayed as grey; when in contact, its color changes to orange. Opacity increases proportionally with the amount of normal force with which the stick is pressed onto the masses, N. How this normal force is controlled is discussed in detail in the

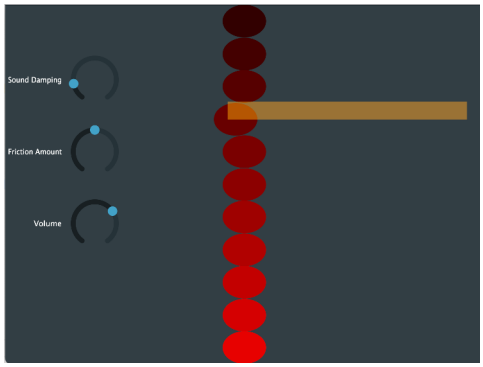


Figure 4: Snapshot of the audio application window. The stick (yellow rectangle) is exciting one of the masses (red circles), which is displaced from its rest position.

Figure 5: 3D Systems Touch X haptic device.

next subsection. When the stick is in contact with one mass, the velocity of its longitudinal movement is mapped to the rubbing velocity v_r. These are the two excitation parameters, as was illustrated in the sketch of the physical model given in Fig. 1. Of course, the stick can only rub one mass at a time, actually the one over which it is placed.

When the stick is lifted, the masses will continue to oscillate with an exponential decay that can be varied via the software knob labeled "Sound Damping", ranging from zero damping, i.e. constant sustained sound, to an amount large enough to remove all the release of the sound.

The second knob in the audio application, named "Friction Amount", controls the friction parameter a whose effect on the model was discussed in Section 2.1. When it is all the way to the left, the friction interaction will be completely smooth and the stick will have no effect on the masses and slide over them. When it is turned all the way to the right, the interaction becomes ideally adhesive and the masses will not slide, conversely they will be dragged by the stick. In this way the stick can be used to pluck the masses, thus creating a sound with a more sharp attack.

The linear range of values obtained from the knobs was mapped to the physical model parameters using a heuristically determined exponential mapping that was found to give a reasonable perception of their effect. Another such heuristic approach was applied by scaling the displacement of each mass before the output to the audio application. This was due to the fact that mass-spring-dampers tuned to lower frequencies produce larger amplitudes than oscillators resonating at higher frequencies, holding an identical excitation. This effect is due to the smaller stiffness of the former.

The last knob, "Volume", controls the global sound volume and is mapped over a logarithmic scale going from -30 dB to 3 dB.

In order to avoid audio clicks and with the intent of having a good dynamic range of the sound, a simple limiter was added, based on the design given in [15]. Finally as an extra safety measure, a DC filter, was also included.

3.3 Control Strategies

Three control strategies were developed for the app with an aim to mimic with increasing accuracy the gestural action of rubbing a mass using a stick.

3.3.1 Mouse

First off, the interaction with the app was based on the universal control device for virtual musical instruments, i.e., the mouse. Incidentally, the action of "rubbing" is somewhat inherent in the dragging of the mouse, so the two gestures are in fact rather comparable.

The position of the mouse is mapped to the position of the center of the stick in the application window. When the mouse is clicked, the stick enters in contact with one mass, which can then be dragged longitudinally. The rubbing normal force, N, can be modified using the mouse wheel. Of course if one does not have a mouse at hand, the trackpad of a laptop can be used instead. In the case of a MacBook trackpad, the equivalent of tilting the mouse wheel is sliding two fingers up & down over it.

3.3.2 Sensel Morph

The second control strategy was implemented using a device named Sensel Morph, a tablet-sized pressure sensitive touchpad which is especially fast and sensitive.

Behind this control, was the idea to include the pressing action as part of the instrumental gesture, an action which can be naturally mapped to the rubbing normal force. With this in place, a more dynamic control of this excitation parameter is available for the user.

As opposed to the mouse control, where a binary mapping for putting the stick in contact with the masses is available, in the case of the Sensel this mapping cannot be explicitly set on and off. It was decided to map the position of the stick to the point where a finger first touches the device. The touch by a second finger signals that the stick must be put in contact with a mass, with normal force dependent on its pressure.

3.3.3 3D Systems Touch X

For the final control strategy, the purpose was not only to simulate the instrumental gesture even further but also include realistic haptic feedback based directly on the physical model, thus reinforcing the multisensory interaction with the audio application. This control could be achieved by using the 3D Systems Touch X haptic device, which is

illustrated in Fig. 5 together with a reference coordinate system, used in the following descriptions. It is a 6-degree of freedom pen-shaped robotic arm, equipped with a series of motors which can provide 3-degree of freedom force feedback.

The position of the tip of the pen in the x-z plane was mapped to the location of the stick in the application window. The movement of the arm in the z direction was bounded by introducing two very stiff haptic x-y planes along the z axis, creating the feeling of hitting a wall. Another such haptic plane was modelled parallel to x-z at elevation y = 0. However this plane is elastic, providing force feedback proportional to the penetration along the y direction, corresponding to negative y coordinate values. When $y > 0$ conversely no feedback is felt, with this situation being mapped to the case when the stick is hovering above the masses. When the pen tip is at y = 0, contact with the masses occurs. The normal rubbing force, N, is then mapped to the force feedback felt in the y direction, coming from the elastic plane. Lastly, the movement velocity of the pen tip along the x direction is mapped to the rubbing velocity v_r.

In order to increase the realism of "rubbing", the friction force resulting from the physical model simulation is mapped to the force feedback in the x direction. In this manner, users can actually feel the stick-slip interaction between the stick and the masses.

4. EVALUATION

A user study was carried out aiming to evaluate the overall experience of using the virtual instrument with the different control strategies. The evaluation followed the guidelines given by Barbosa et al. in [16], with a focus on clearly stating the goal, methods and criteria used. As for the audio output signal, it was given by the displacement of the masses, since the velocity option was not developed at that time.

The goal was to compare the control strategies and investigate whether mimicking the instrumental gesture benefits the experience. Furthermore, there was a desire to collect feedback from users in order to improve the audio application and the interaction. This was achieved by means of both quantitative and qualitative methods, namely by: (1) a questionnaire composed of 7 statements, each related to a single attribute/criteria, which could be rated on a 7-point Likert scale, from "Strongly Disagree" to "Strongly Agree". The statements and their associated attributes are given in Table 1. They are formulated in such a way that responses on the upper part of the scale (agreement) indicate positive opinions with respect to the attributes. (2) A qualitative interview was carried out after the subjects completed the questionnaire. Here, the focus was on investigating whether there was an understanding of what the different controls of the application did and how they affected the sound, i.e. turning the different knobs or rubbing the masses with different velocities and pressure. Lastly, their feedback regarding possible improvements to the system was noted.

Attribute	Statement
1. *intuitiveness*	The interaction with the objects was intuitive.
2. *playability*	I obtained sounds as by my intentions.
3. *enjoyment*	The interaction was fun.
4. *expressiveness*	I obtained enough types of sounds.
5. *difficulty*	It was easy to obtain the same sound twice.
6. *realism*	The interaction was realistic.
7. *precision*	The interaction was precise.

Table 1: The questionnaire items with corresponding anchors of the 7-point (1-7) Likert scale (Strongly Disagree - Strongly Agree).

4.1 Participants

A total of 14 participants took part in the user study, mostly comprising of students and staff at University of Udine in Italy. All but 2 persons had experience with playing music instruments, albeit not in any professional manner. Only one person had considerable experience using virtual instruments.

4.2 Procedure and Task

Each session started with a verbal introduction to the participants about the audio-haptic application and the physical model behind it. Showing the sketch of the system illustrated in Fig. 1 to the participants was found to be helpful in this regard.

Three instances of the app were opened, overall fitting on a large monitor. Each was respectively controlled by a device among those presented in Section 3.3. A brief tutorial on how to use the devices was carried out, during which the auditory feedback was turned off. This avoided users to experience how they could produce sounds of different quality instead of letting them explore the interface by themselves. The visualisation of the oscillation of the masses, however, was left on, to clearly illustrate the app's response to control. Users were not instructed about the haptic feedback from the Touch X, as it was desired they discover it themselves.

After the briefing, the audio was turned on via a pair of Genelec 8020 active loudspeakers, by means of an RME Babyface PRO sound interface. Subjects were then free to use any of the three apps and their respective controllers in whichever order they prefer including the possibility to move across them more than once. Before they started, they were encouraged to experiment with the different knobs and try to get a feeling of what such controls did. Every participant's activity was observed and notes were taken. Here, using loudspeakers as opposed to headphones was useful, as the experimenter could, among other things, identify different sonic preferences of the subjects or infer pressures applied to the stick via the auditory feedback.

Once they experimented with all three controllers, they were requested to fill out the questionnaire. For each ques-

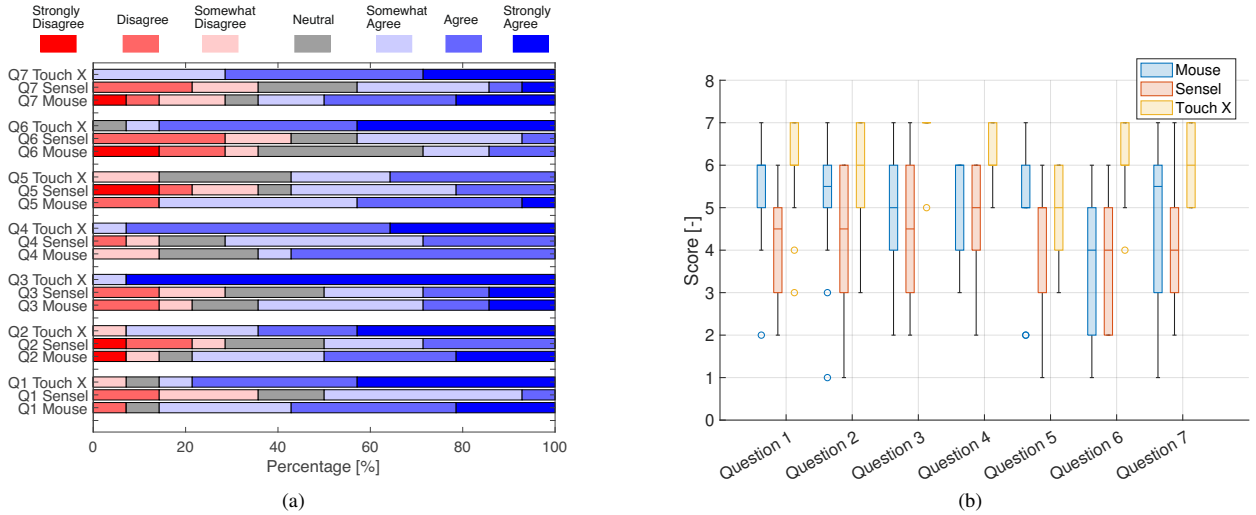


Figure 6: (a) Visualisation of the quantitative data illustrated as stacked bars. The three control devices: Mouse, Sensel and Touch X are grouped together for each of the 7 questions: Q1 to Q7. (b) Boxplot visualising the results related to the 7 questionnaire statements for the three control devices.

tion, they would provide a score for each controller, i.e., they went through the questions only once. This strategy highlighted the comparison between the devices, as the scores were more relative to each other. Once the questionnaire was filled, participants were interviewed about their experience.

Every session lasted between 10 and 30 minutes, depending on how long the users chose to use the apps, as there was no fixed time limit for that.

4.3 Results

The data from the questionnaire are presented in this section, followed by the qualitative findings from the interview.

4.3.1 Quantitative Data

An overview of the data can be provided by a stacked bar plot, as illustrated in Fig. 6a. This plot gives an idea of the relationships among such data and allows for an informal comparison between the devices. It can be seen that most responses for the control strategy using the Touch X haptic device fall in the positive Likert scale range (”Agree” to ”Strongly Agree”). Results are more mixed in the case of the two other controllers.

Furthermore, the Likert scale data can be treated as ordinal and a good indication on how the results are spread out can be achieved by box plot visualisation, as shown in Fig. 6b. This gives an assessment of the results in terms of central tendencies (median, mode), interquantile ranges and minimum/maximum ratings. Here again, one can see that the Touch X consistently scores better than the two other devices. When it comes to *enjoyment*, the users almost unanimously ”Strongly Agree” that the interaction was fun, with only a single score of ”Agree”. A similar positive distribution of scores is found with respect to *expressiveness*, *realism* and *precision*, with each of these attributes receiving answers solely in the positive range. The lowest scores for the Touch X were found for the *difficulty* statement, which is not surprising considering that none of the participants had used such a device before and a learning curve for its use is to be expected.

A somewhat unexpected result is that the mouse was generally preferred over the Sensel. It seems that the benefit for participants to rely on an intuitive dynamic pressure control is not enough, at least at first exposure, to overcome the familiarity they already have with the mouse and, at least to some extent, with trackbars aboard laptops. In this sense, the dragging gesture afforded by a mouse may have found an ideal match with the experimental task. Consequently, subjects found the mouse to be more intuitive, and easier to play with. One additional reason for this preference over the Sensel could be that using one finger to move the stick and two fingers to rub the masses could be difficult to learn, in spite of the availability nowadays of two-finger commands in several trackbars e.g. for scrolling windows.

As a matter of fact, the mouse scored best with respect to the *difficulty* attribute, suggesting that it was the easiest to use of the three devices. Again, this score testifies its ubiquity in computer setups.

4.3.2 Qualitative Data

This section summaries the insights obtained by observing participants during the task, and from the interviews they gave after filling out the questionnaires.

First off it is worth mentioning that all participants could easily explain and identify the effect of the ”Sound Damping” knob, with answers such as: ”[it] controls the release of the sound” or ”how long it [the mass] continues to vibrate”. When it comes to the ”Friction Amount” knob users were less certain, but did have some correct intuitions regarding its mechanism. For instance one said that it controls ”how strong you have to press to make a sound”, while another mentioned that it ”changes the perceptions of hardness or viscosity. . . like stirring a soup”. Indeed,

with low friction values one needs to press harder to excite the masses, but with high values, corresponding to sticker interactions, this is not the case. Here, rubbing does indeed feel more viscous (one of the definitions of the word being "having a thick or sticky consistency"). Users mentioned that using the Touch X haptic device did help them better understand this parameter. After further explanation users could easily identify ranges for the knob where they could feel the stick-slip feedback. One reason why perhaps they were not so aware of this mechanism at first try, is that most of them focused on rubbing the masses in an effort to produce musical sounds, which is achieved at lower rubbing pressures, where the magnitude of the friction force feedback is consequently smaller. Another trend in the interviews was that there were plenty of references to the sounds of friction, with one user comparing the sound directly to that of bowed instruments like the violin, while another saying it "sounds like when rubbing a wet glass".

Some users spent a considerable amount of time with the applications and were noticeably experimenting with the devices and the sounds, while others were quite more reserved. In fact one of the persons with no experience with any musical instrument had very slow and unnatural movements, which resulted in unmusical sounds, seemingly reflecting a difficult interaction. This is a statement to resemblance of physics-based sound synthesis with real-world instruments, in the sense that one needs to learn the virtual instrument in order to fully enjoy the experience.

5. CONCLUSION

In this paper, a friction based physical modelling sound synthesis application was introduced, for which three different control strategies were developed. With these, the aim was to differently replicate the gestural action of rubbing in terms of control. We assumed that preserving the link between the excitation gesture of the physical system and the resulting sound would enhance the quality of the experience. This expectation was evaluated via a preliminary user study which showed a clear preference for controlling the app via the 3D Systems Touch X haptic device. The device not only mimicked the instrumental gesture, but also provided haptic feedback in terms of pressure response and the friction resulting from the physical model. This result is another indication of the importance of implementing intuitive interfaces for virtual instruments, and how haptic feedback plays a key role in this process.

6. REFERENCES

[1] C. Cadoz, "Instrumental gestures and musical composition," in *ICMC*, 1988.

[2] M. S. O'Modhrain, "Playing by feel: incorporating haptic feedback into computer-based musical instruments," Ph.D. dissertation, Stanford University, 2001.

[3] S. Willemsen, R. Paisa, and S. Serafin, "Resurrecting the tromba marina: A bowed virtual reality instrument using haptic feedback and accurate physical modelling," in *Proceedings of the 17th Sound and Music Computing Conference*, 2020.

[4] F. Fontana, R. Paisa, R. Ranon, and S. Serafin, "Multisensory plucked instrument modeling in unity3d: From keytar to accurate string prototyping," *Applied Sciences*, vol. 10, p. 1452, 2020.

[5] G. Wang, "Designing smule's ocarina : The iphone's magic flute," in *Proceedings of the 2009 International Conference on New Interfaces for Musical Expression*, 01 2009.

[6] E. Y. Huynh, "Bowed plates and blown strings: Odd combinations of excitation methods and resonance structures impact perception," Master's thesis, Department of Music Research, McGill University, Canada, 2019.

[7] M. G. Onofrei, S. Willemsen, and S. Serafin, "Real-time implementation of a friction drum inspired instrument using finite difference schemes," in *Proceedings of the 24th International Conference on Digital Audio Effects (DAFx20in21)*, 2021.

[8] S. Sinclair, G. Scavone, and M. Wanderley, "Audio-haptic interaction with the digital waveguide bowed string," in *Proceedings of the 2009 International Computer Music Conference, ICMC 2009*, 2012.

[9] S. Bilbao, *Numerical Sound Synthesis*. John Wiley and Sons, Ltd, 2009.

[10] S. Willemsen, S. Serafin, S. Bilbao, and M. Ducceschi, "Real-time implementation of a physical model of the tromba marina," in *Proceedings of the 17th Sound and Music Computing Conference*, 2020.

[11] P. Christensen, D. Overholt, and S. Serafin, "The daïs: A haptically enabled nime for controlling physical modeling sound synthesis algorithms," in *Proceedings of the 2020 International Conference on New Interfaces for Musical Expression*, 2020.

[12] S. Willemsen, "The emulated ensemble: Real-time simulation of musical instruments using finite-difference time-domain methods," Ph.D. dissertation, Aalborg University, 2021.

[13] M. G. Onofrei, "Audio-haptic frictional excitation of a mass-spring-damper physical model," https://www.youtube.com/watch?v=raLtqhdOYOM, Youtube.

[14] M.G.Onofrei, "Bowed mass-spring-damper apps," https://www.dropbox.com/sh/xclkp9ukz9qryn2/AAAk5XXyn0iU5E1Po9LU5x1Ma?dl=0, Dropbox.

[15] P. Dutilleux and U. Zölzer, "Chapter 5 - nonlinear processing," in *DAFX - Digital Audio Effects*. John Wiley & Sons, 2002.

[16] J. Barbosa, J. Malloch, M. Wanderlay, and S. Huot, "What does "evaluation" mean for the nime community?" in *Proceedings of the 2015 International Conference on New Interfaces for Musical Expression*, 2015.

1e0a: A COMPUTATIONAL APPROACH TO RHYTHM TRAINING

Noel Alben
Georgia Institute Of Technology
noelalben@gatech.edu

Ranjani H.G.
IEEE Member

ABSTRACT

We present a computational assessment system that promotes the learning of basic rhythmic patterns. The system generates multiple rhythmic patterns with increasing complexity within various cycle lengths. For a generated rhythm pattern the performance assessment of the learner is characterized with summary statistics estimated from the onsets of the learner's performance and provided as feedback. When performance assessment feedback is within certain error bounds, the system proceeds to generate a new pattern of increased complexity. The system thus mimics a learner-teacher relationship as the learner progresses in their feedback-based learning. The choice of progression within a cycle for each pattern is determined by a predefined complexity metric. This metric is based on a coded element model for the perceptual processing of sequential stimuli. A model earlier proposed for a sequence of tones and non-tones, is now extended for onsets and silences. This system is developed into a web-based application and provides accessibility for learning purposes. Analysis of learner performance assessments from a pilot user study shows us that the complexity metric is moderately indicative of the perceptual processing of rhythm patterns and with further enhancements can be used for rhythm learning and future syllabus generation of rhythm training tools.

1. INTRODUCTION

Learning to play a musical instrument is a complex process that requires the acquisition of many interlinked skills such as tone, intonation, note accuracy, timing, rhythm precision, clear articulation, and dynamic variation [1]. An acknowledged important skill is the understanding of rhythm, timing, and rhythm precision.The word rhythm is used to refer broadly to all temporal aspects of musical works, and more specifically the duration and durational patterns of musical notes [2], [3]. Over the course of their musical education and understanding of rhythmic concepts one gradually starts to develop an intuition towards family of rhythmic patterns and similarity between different patterns, this intuition and understanding is usually the first step towards mastering the art and developing the skill set.
Two main contexts within which formal musical learning takes place are the classroom, through playing and interacting with a teacher, and at home, practicing alone [1]. From the perspective of a drum kit or percussive instrument, a typical classroom setting involves the teacher and a learner. The teacher decides what the learner must practice and the teacher's feedback helps them achieve accuracy in timing and quality in performance. The learner's practicing habits at home aid in reaching performance goals. A software application cannot substitute a music teacher but it can be a supplement in the learning process, especially in the performance assessment of daily practice at home [1].

Copyright: © 2022 Noel Alben et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

1.1 Related Literature

Most assessment methods are either the assessment of learning, used to measure competence, or the assessment for learning used to enhance learning [4], [5], [6]. The rapid development of music technology over the last few decades has dramatically changed the way that people interact with music. In [7], the authors explore the use of Music Information Retrieval (MIR) techniques in music education and their integration in learning software. They outline Song2See as a representative example of a music learning system developed within the MIR community. Songs2See is a music game developed based on pitch detection, sound separation, music transcription, interface development, and audio analysis technologies [8]. Despite advancements in these sound analysis technologies, Eremenko V, Morsi A, et al. in [1], revealed that the performance assessment tools present in these applications are still not reliable and effective enough to support the strict requirements of a professional music education context. In [1], the authors review some of the work done in the practice of music assessment and present a proof of concept for a complete assessment system for supporting guitar exercises. Furthermore, they identify the challenges that should be addressed to further advance these assessment technologies and their useful integration into professional learning contexts, particularly those that will help learners to plan, monitor, and evaluate their learning progression. They also highlight the importance and requirement of presenting the assessment results to a learner in a way that promotes learning [1]. Similarly, Lerch, A, et al. in [9] surveyed the field of Music Performance Analysis (MPA) from the perspective of performance assessment and its ability to provide insights and interactive feedback by analyzing and assessing the audio of practice sessions. Furthermore, they highlight the necessity for individual assessment and curriculum in the context of a music education tool. The work of Chih-Wei Wu and Alexander Lerch in [10], presents an unsupervised feature

learning approach to derive features for assessing percussive instrument performances amongst a large data of audition performances, mimicking a judge in this scenario. Specifically, they proposed using a histogram-based input representation to sparse coding to allow the sparse coding to take advantage of temporal rhythmic information. Various (commercially available) solutions exist specifically for rhythm education and performance assessment. Examples for rhythm tutoring applications are Perfect Ear [11], Rhythm Trainer [12], and Complete Rhythm Trainer [13]. These form a class of mobile applications that help learners practice and assess their rhythm skills by following a set curriculum presented in each of these applications and touchscreen-based feedback for performance assessment. Other apps such as SmartMusic [14], Yousician [15], Music Prodigy [16], and SingStar [17] are available for music education purposes however, they do not have a dedicated rhythm training section. Also, none of the previous works address musical rhythm complexity measures as a feature to aid in a learner's progression. Evaluating various rhythm complexity measures based on how accurately they reflect human-based measures were studied by Eric Thul in his thesis [3]. Although his results suggest that none of the measures accurately reflect the difficulty humans have performing or listening to rhythm, the measures do accurately reflect how humans recognize a rhythm's structure.

1.2 Our Contribution

In this work, we consider the individual assessment of learning a particular rhythmic pattern at a fixed tempo. We present the proof of concept for a system that will assess a learner's audio recording and provide performance feedback to them. We will only focus on assessing the learner's temporal proficiency, keeping the performance parameter categories of tempo, dynamics, pitch, and timbre fixed. Our main contribution is employing a pre-defined complexity metric found in [3] to enable the learner to progress from one rhythm pattern to the next, similar to a teacher who decides what the learner must practice.
This manuscript is organized as follows, we first define the terms, notations, and conventions to be held throughout that will help assess the performance of an individual. Followed by the proposed system for the assessment where we go over the subsequent blocks that make up the assessment system and understand the coded element model used to define the complexity metric. Finally, we discuss the pilot user study results and evaluation of this system through our web application platform and a conclusion with the future enhancements and improvements to the system.

2. TERMINOLOGY

2.1 Definitions

We first provide a brief background of some of the terms used to define a rhythmic pattern.
In this work, we consider the fundamental and individual time unit for a rhythmic pattern to be a *pulse*. Pulse is defined to be one of a series of regularly recurring, precisely equivalent stimuli [18]. The duration of a rhythmic pattern

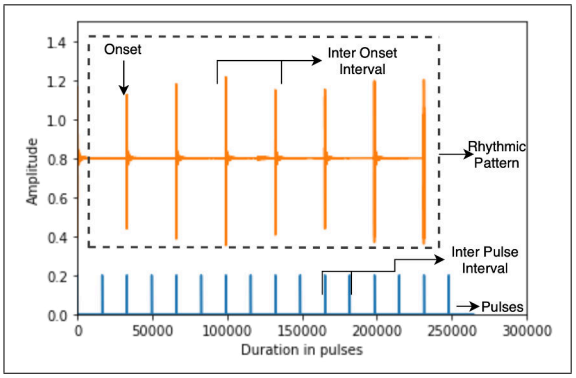


Figure 1. An illustrative diagram of the terms used in this work.

is composed of a number of pulses. Tempo refers to the rate at which the musical pulses are generated and it also controls the number of pulses for a given duration. For our proposed system the total duration of a rhythmic pattern is determined by the total number of samples present in the audio signal. The uniform number of samples or duration, between two subsequent pulses, is defined to be the inter-pulse interval (IPI). The IPI depends on the sampling frequency and the tempo used for the rhythmic pattern. Figure 1 shows a duration of 6.80 seconds with pulses generated at a fixed Inter Pulse Interval (IPI), determined by a tempo of 160 Pulses Per Minute (PPM).
To generate a rhythmic pattern, we consider instantaneous attacks on the marked time units (pulses) i.e., striking a drum, clapping, or tapping. The instantaneous attacks are defined as onsets. The duration that is marked between two onsets is defined as the inter-onset interval (IOI). The IPI, onsets, and the IOI together create a rhythmic pattern as shown in figure 1.
A cycle determines the number of pulses beyond which the rhythmic pattern begins to repeat itself. The cycle represents a structure for the pulses, and this structure is realised by the rhythmic pattern. For example, we use 7 to indicate a repeating rhythmic pattern of 7 pulses. Rhythm patterns can be composed in any predetermined cycle number, for this article we will only be following cycles of 4,3,5,7, and 16.

2.2 Representation

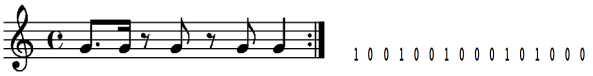
We will now introduce the rhythm pattern representation used in the proposed system with the help of a famous Afro-Cuban rhythm, the clave son, shown in figure 2 (a).

(a) (b)

Figure 2. (a) The clave son rhythm in music (stave) notation. (b) The clave son rhythm in binary notation.

Figure. 2 (b) shows the clave son rhythm as a binary string, a notation that describes the position of the onsets

and silences. This notation is used in the domain of computer science [19]. Each digit represents a single pulse, with the total number of digits indicating the cycle length. The value of the digit itself indicates whether, in a particular rhythmic pattern, a onset occurs on this pulse (1) or not (0). This representation enables faster computation and integration of rhythm patterns into our system. This representation of a rhythm pattern henceforth will be referred to as the binary representation.

2.3 Naming 1e0a

The name *1e0a* for the proposed system is derived from a mixture of the binary representation with its 1's and 0's and a western rhythm counting technique used by a large majority of music educators, called the Harr system. The system that Harr used assigned a set of numbers and syllables. For a cycle of length 4, numbers are assigned to onsets that occur at the beginning of the rhythmic pattern's duration and its repeats. Syllables are assigned to those onsets that fall everywhere else. The syllables remain consistent for all the pulses with only the number for each repeat changing. For example, the rhythm [11111111] would be read, *1 e & a 2 e & a* [20]. In our work, we adopt the same system, we use the name ***1e0a***.

3. PROPOSED SYSTEM

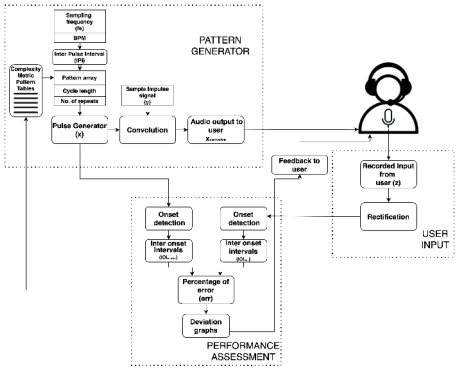


Figure 3. Block diagram of the proposed system for *1e0a*.

Figure 3 shows the block diagram of our proposed system for rhythm training and progression in learning. This system consists of two distinct blocks: (1) The rhythm pattern generator with the complexity metric, and, (2) the performance assessment of the learner input with feedback.

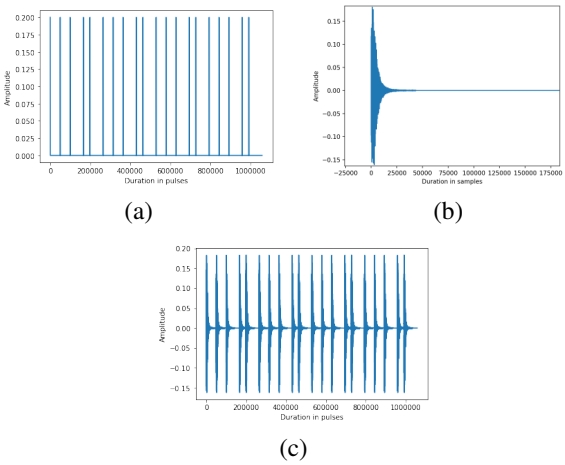
3.1 Pattern Generator

3.1.1 Pattern Generation

The pattern generator as shown in figure 5 begins with a pulse generator function. This function takes input parameters that determine the cycle length, pattern array, tempo (PPM), and the total number of repeats, required to generate a collection of pulses whose inter-pulse interval (IPI) is determined by the equation 1.

$$IPI = (60/PPM)/(1/f_s) \quad (1)$$

PPM represents the pulses per minute of the generated collection of pulses, f_s represents the sampling frequency and IPI represents the fixed inter-pulse interval.
The pattern array in binary representation determines the location of onsets and the number of silences to be incorporated into the duration of pulses. We also propose the selection of pattern arrays for generating subsequent sequences using complexity metrics detailed in section 3.1.2. For the clave son: [1 0 0 1 0 0 1 0 0 0 1 0 1 0 0 0] , at 160 PPM, with 4 repeats, at an f_s of 44100. The function generates an array of pulses as shown in figure 4(a).



(a) (b)

(c)

Figure 4. Pattern Generator plots. (a) Pulse Generator Output; (b) Hi-Hat Drum sample impulse signal; (c) Convolution result of (a) and (b).

The next stage of the pattern generator block adds tonal quality to the output of the pulse generator, by convolving the pulse generator output with an appropriate drum sample impulse signal. The drum sample impulse signal is a short recording of an onset played using a particular drum sound. The drum sound selected must have an instantaneous attack and a short decay as shown in figure 4(b). The convolution process requires the loaded sample impulse to have the same f_s as the pulse generator output. For a pulse generator output and drum impulse, figure 4(c) shows the output of the convolution function. We make use of the Discrete Fourier Transform convolution property [21]. This allows our result to have the same number of samples as the pulse generator output. We use the Fast Fourier Transform (FFT) [21] algorithm to achieve this result. The resultant clave son rhythm from the pattern generator block, at 160 PPM and 4 cycles, is an audio signal whose plot is shown in figure 4(c).

3.1.2 Complexity Metric

Various measures of musical rhythm complexity have been studied in [3]. Rhythm complexity measures have been studied as:
Rhythm syncopation measures that use metrical weights determined from the metrical hierarchy to find syncopated

onsets in a rhythm [[19], [22]].
Pattern matching measures which chop up a rhythm as determined by the levels of the metrical hierarchy and search each sub-rhythm for patterns that indicate the complexity [[23], [24]].
Distance measures, which measure how far away a rhythm is from a more simple rhythm composed of beats [19], [25], [26]].
Shannon's information entropy [27], [28], which are mathematical measures which are based upon **Shannon's information entropy** uncertainty equations given in equation 3, 4, 5.
We estimate the rhythmic complexity measure using information entropy approach by estimating the entropy for a particular rhythm pattern. Vitz and Todd in [28], present a method that can be used to calculate the complexity of cyclic binary patterns. Their work defined that the complexity measure of a certain periodic or repeating pattern is the sum of the total amount of uncertainty evaluated in processing the stimulus sequence. In [28], the authors achieved this by proposing a set of rules (axioms) describing a perceptual coding process. As the method implies an emphasis on coded elements, it is referred to as a coded element processing system (CEPS) [3].
We consider this model for our application due to its algorithmic approach to measure rhythm complexity, thus making it viable for coding, as against rhythm syncopation models in [19]. CEPS also has an improved ability to distinguish rhythm patterns within a cycle based on complexity as compared to pattern matching [23], [24] and distance measures [19], [25], [26].
The axioms in [28], proposed originally for tones and non-tones, break down sequences into code levels. In our work, we extend it to rhythmic patterns, composed of onsets and silences. Each code level successfully codes the pattern from single elements to larger elements which continues until the entire pattern is constructed. To calculate the complexity of the rhythmic pattern, we measure two uncertainty values at each Code Level in the algorithm as proposed in [28]. Uncertainty in this case means information entropy. The uncertainty values are the maximum uncertainty, H_{max}, and the joint uncertainty, H_{joint} represented in equations 2 and 3.1.2 respectively. Let X, Y, Z, W represent random discrete variables.

$$H_{max}(X,Y,Z,W) = H(X) + H(Y) + H(Z) + H(W) \tag{2}$$

$\mathrm{H}_{joint}(X,Y,Z,W) = H(X) + H(Y|X) + H(Z|X,Y) + H(W|W,Y,Z)$ where,

$$H(X) = -\sum_{x \in X} p(x) log_2 p(x) \tag{3}$$

$$H(X,Y) = -\sum_{x \in X}\sum_{y \in Y} p(x)p(x,y) log_2 p(x,y) \tag{4}$$

$$H(X|Y) = -\sum_{x \in X}\sum_{y \in Y} p(y,x) log_2 p(y/x) \tag{5}$$

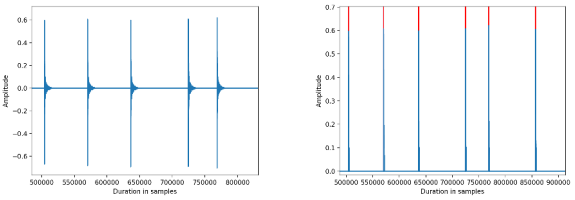
Let the value H^k be the sum of H^k_{max} and H^k_{joint} at Code Level k. The total sum of H at each Code Level represents the complexity of the pattern. This value is termed H_{code}.

$$H_{code} = \sum_{k=1}^{n} H^k = \sum_{k=1}^{n} H^k_{max} + H^k_{joint} \tag{6}$$

The complexity measure for $[1001001000101000]$ is calculated to be **20.5538**. The complexity measures are precalculated for all the combinations of patterns within the cycle lengths of 4,3,5, and 7. Then, the patterns are ordered based on increasing complexity as shown in $sheets$ [1] . We observed that for patterns that are a shifted version of another, where 1's (onsets) replace the 0's (silences), and vice versa, the complexity measures are identical. For example, $[1100]$ and $[0011]$ have a complexity measure of 5.0. To tackle this, we considered those patterns that begin with a 0 as off-beat patterns and assumed that they present a higher challenge to learners. Hence, we incremented a constant value to the complexity measures for all those patterns that begin with a 0 (silence). The constant value within a cycle was determined by the highest complexity measure for the patterns that begin with 1 (onsets), for that cycle length. This can be observed in $sheets$[1].
However, this complexity metric does not account for the complexity that arises with changes in tempo. Therefore in this work, we focus on studying the performance assessment at a constant PPM.

3.2 Performance Assessment

Once the learner listens to the generated pattern their performance is recorded and loaded onto the system. We rectify the recorded audio signal (z) with a threshold, $z_{threshold} = z_{max}/4$, to improve the results of the onset detection function and to account for low magnitude noise in the live recording environment.



(a) Recorded input. (b) Onsets from rectified inputs.

Figure 5. Plots of onset detection in one cycle.

Any onset detection function can be used to extract the onsets from the rectified recorded input. We used the onset detection function defined by the Librosa python library [29]. A sample of the recorded input and the extracted onsets are shown in figure 5.

Inter onset intervals are calculated from onsets estimated from both recorded (IOI_{input}) and generated signals ($IOI_{generated}$).

[1] https://docs.google.com/spreadsheets/d/e/2PACX-1vTSDSqil264H6-M2kLdQUuD6VYnRqYPQePhKl8C_STD1j-54rFdhbLag-ep_15ggiHpo8-DOpJWyQ-O/pubhtml

```
[0.15 −0.04 0.01 −0.14 0.01 ]
[−0.04 0.15 0.01 −0.14 0.01 ]
[−0.04 −0.04 0.01 0.15 0.01 ]
[−0.04 −0.04 0.01 0.15 −0.14 ]
```

Figure 6. $4X5$ Matrix of error deviation result.

The deviation: $IOI_{generated} - IOI_{input}$, for each inter onset interval is calculated to be the error in temporal performance at each onset. If the user inputs different number of onsets as compared to the generated pattern, we pad the audio if it's lower or truncate the audio if it's longer.

The error is recorded in a matrix format where the number of columns is equivalent to the number of onsets in a cycle and the number of rows is determined by the total number of cycles. For the son clave rhythm, there are 5 *1's* (onsets) and 11 *0's* (silence) played for 4 cycles, resulting in a $4X5$ matrix as shown in 6.

From the error matrix, two sets of averages are calculated. One set of average values represent the deviation of each inter onset interval within a cycle (average of all the values in one column of the error matrix). This results in a $5X1$ average onset error array. This average onset error array is plotted on a graph with the horizontal axis representing each onset and the vertical axis representing the percentage of deviation (-100 to 100) This is shown in figure 7(left). This result indicates to the learner their timing error at each onset. The negative error implies slowly playing speed and the positive indicates fast playing speed.

The next set of average values represent the error deviation of a learner across the cycles (average all the values in one row of the error matrix) which results in a $4X1$ average result array. This result is plotted on a graph with the horizontal axis representing the number of cycles and the vertical axis representing the deviation (-100 to 100) shown in figure 7(right). This result gives the learner information on their consistency throughout the performance and indicates if the timing error is consistent across cycles.

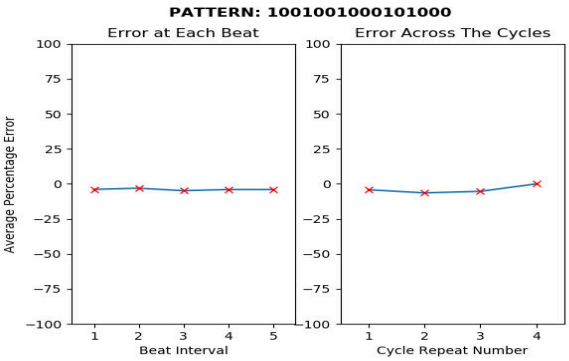


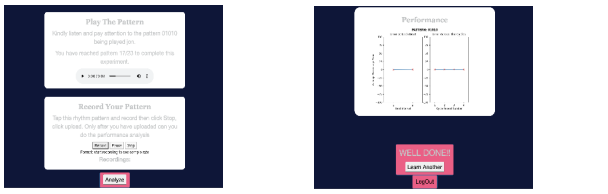
Figure 7. Deviation graph results of the performance assessment block.

The graphs in figure 7 give the learner comprehensive performance feedback concerning their timing and consistency. We achieve this through the graphical representation wherein, a positive deviation (above 0 line) in figure 7 (left) infers that the learner is rushing or playing the beat faster than intended and a negative deviation (below 0 line) infers that the learner is dragging or playing the beat slower than intended. Each onset in figure 7 (left) is marked by an **x**.

4. 1E0A - THE WEB APPLICATION

For improved accessibility to this system to aid in learning, we developed a web application hosted at `https://1e0a.noelalben.com`

Our goal was to implement the proposed system with a simple user interface and a server to collect learners' data. We have been able to achieve both by using Flask a micro web framework written in Python [30]. The website was hosted on AWS [17] with certbot SSL certification [31].



(a) Learning platform (b) Performance Assessment results

Figure 8. *1e0a* Web Application.

4.1 Data collection

Every new learner must register with the website and provide consent to use their registration information and performance assessment results for research purposes. Post-registration the learner is given a login id and password. After logging in to the system, they enter the learning platform as shown in figure 8 (a). They must listen to the generated pattern and record their respective performance. The *Analyze* button conducts performance assessment on the learner's recording and takes them to the results as shown in figure 8 (b). If they choose to progress in their learning by clicking *Learn Another*, the system will generate the next pattern based on the complexity metrics defined in section 3.1.2 and take the learner back to the learning platform. When they log out, the most recent generated pattern is stored into the database and they can continue where they left, upon logging in again.

5. PILOT STUDY

5.1 Setup

To assess and quantify the effectiveness of our rhythm learning system, each learner on the 1e0a web application was required to perform and record 23 distinct patterns over 4 different cycle lengths of (4,3,5 and 7), at a constant PPM of 160. The learners were asked to use a desktop/laptop to access the web application and to maintain a constant interface to perform the rhythmic patterns (clapping or a single percussive instrument). They were not allowed to use accents in their performance nor take breaks while performing a given pattern. Throughout the experiment, the application recorded the number of attempts

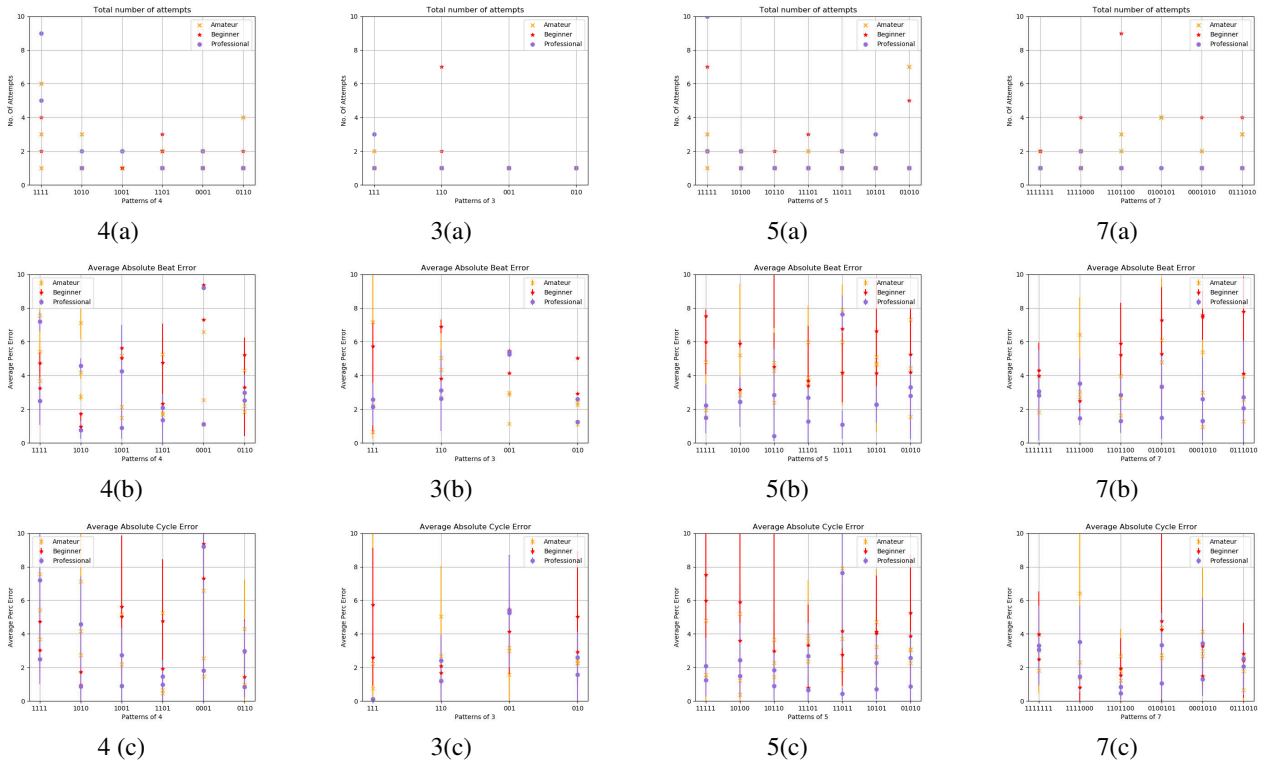


Figure 9. Graphical analysis of performance assessment for 7 learners categorised by professional(•), amateur(×), and beginner(⋆) on the 1e0a web application at each cycle. (a)Top Panel: Total number of attempts taken by learners. (b) Center Panel: Absolute average beat error in performance. (c) Bottom Panel: absolute average error over the cycle in performance.

each learner takes to achieve performance assessment results within the error bounds presented in section 3.2. Furthermore, we stored the performance assessment results of the learner for each rhythmic pattern where their results were within the error bounds. The results of 7 learners are shown in fig 9. Each learner is categorized based on their years of experience with music education: (2) Professionals (>10 years), (3)Amateur (>2 years), and (2) Beginners(0-2 years).

5.2 Results

From the resulting graphs in figure 9, it can be observed that compared to an amateur or a beginner, for any given pattern the professional requires less number of attempts to achieve performance accuracy within the error bounds. This is reflective of a learning system where beginners make more mistakes and progressively improve in their performance as they spend more time on the application. We also observe that the learners were able to correct their mistakes and achieve performance accuracy within the error bounds after a sufficient number of attempts. This indicates that the performance assessment feedback of the system aids the learner in understanding and correcting their performance to achieve accuracy. Also, it can be observed that across different patterns within the cycle there exists complexity because the number of attempts taken by the learner increases with the progression of patterns within a cycle. However, this does not directly reflect the difficulty that learners have while performing or listening to the rhythm; but does accurately reflect how beginners and amateurs struggle to recognize a rhythm's structure as the complexity increases. For this analysis we did not look into those attempts where the user made mistakes due to the constraints in storage within the application.

6. CONCLUSION

In this article, we introduce *1e0a*, the proof of concept for a rhythm learning and assessment system that helps learners progress in their learning with a new pattern generated based on a pre-defined rhythm complexity measure. We describe the rhythm complexity measure that was used and present the methodology for rhythm performance assessment. The pilot results of the proposed system show that we can successfully test a learner's proficiency on rhythm patterns over a single PPM and monitor their progression within a cycle of patterns of increasing complexity values. This system does not test the learner's proficiency across different tempos neither does it test for mixed cycle complexity.

The future work is aimed at perceptual evaluation on the pedagogical significance of presenting the rhythms progressively using this complexity metric. We are also fo-

cused on building a rhythm generative algorithm using relevant rhythm similarity metrics to obtain a robust syllabus that extends to accented notes and mixed cycle length. Furthermore we would like to improve the system as a learning tool and extend it's capabilities in onset detection, noise removal and user interaction.

7. REFERENCES

[1] V. Eremenko, A. Morsi, J. Narang, and X. Serra, "Performance assessment technologies for the support of musical instrument learning," 2020.

[2] J. London, "Rhythm in l. macy, editor, grove music online. oxford university press," 2008.

[3] E. Thul, "Measuring the complexity of musical rhythm," 2008.

[4] R. Mantie, "The philosophy of assessment in music education," in *The Oxford Handbook of Assessment Policy and Practice in Music Education, Volume 1*. Oxford University Press, 2019, p. 33.

[5] M. C. Schneider, J. S. McDonel, and C. A. DePascale, "Performance assessment and rubric design," in *The Oxford Handbook of Assessment Policy and Practice in Music Education, Volume 1*, 2019.

[6] B. C. Wesolowski, "Understanding and developing rubrics for music performance assessment," *Music Educators Journal*, vol. 98, no. 3, pp. 36–42, 2012.

[7] C. Dittmar, E. Cano, J. Abeßer, and S. Grollmisch, "Music information retrieval meets music education," in *Dagstuhl Follow-Ups*, vol. 3. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[8] E. Cano, C. Dittmar, and S. Grollmisch, "Songs2see: learn to play by playing," in *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*. Miami USA, 2011, pp. 2231–2240.

[9] A. Lerch, C. Arthur, A. Pati, and S. Gururani, "Music performance analysis: A survey," *arXiv preprint arXiv:1907.00178*, 2019.

[10] C.-W. Wu and A. Lerch, "Learned features for the assessment of percussive music performances," in *2018 IEEE 12th International conference on semantic computing (ICSC)*. IEEE, 2018, pp. 93–99.

[11] "Crazy ootka software ab, perfect ear; version: 3.8.71," last accessed 06/02/2022. [Online]. Available: https://play.google.com/store/apps/details?id=com.evilduck.musiciankit

[12] Demax, "Rhythm trainer; version:0.2104261849," last accessed 06/02/2022. [Online]. Available: https://play.google.com/store/apps/details?id=ru.demax.rhythmerr

[13] S. Dupont, "Complete rhythm trainer; version:1.3.10," last accessed 06/02/2022. [Online]. Available: https://play.google.com/store/apps/details?id=com.binaryguilt.completerhythmtrainer

[14] I. MakeMusic, "Smartmusic,april 2019," last accessed 06/02/2022. [Online]. Available: https://www.smartmusic.com

[15] Y. Oy, "Yousician, april 2019," last accessed 06/02/2022. [Online]. Available: https://www.yousician.com

[16] I. d. M. P. The Way of H, "Music prodigy, april 2019," last accessed 06/02/2022. [Online]. Available: http://www.musicprodigy.com

[17] S. I. E. Europe, "Singstar, april 2019," last accessed 06/02/2022. [Online]. Available: http://www.singstar.com

[18] G. W. Cooper and L. B. Meyer, "The rhythmic structure of music. the university of chicago press, 1960."

[19] G. Toussaint, "A mathematical analysis of african, brazilian, and cuban clave rhythms," in *Bridges: Mathematical Connections in Art, Music, and Science*, 2002, pp. 157–168.

[20] B. A. Janssen, *A preliminary comparative study of rhythm systems employed within the first-year college aural skills class*. Kansas State University, 2017.

[21] J. G. Proakis, *Digital signal processing: principles algorithms and applications*. Pearson Education India, 2001.

[22] H. C. Longuet-Higgins and C. S. Lee, "The rhythmic interpretation of monophonic music," *Music Perception*, vol. 1, no. 4, pp. 424–441, 1984.

[23] J. Pressing, "Cognitive complexity and the structure of musical patterns," in *Proceedings of the 4th Conference of the Australasian Cognitive Science Society*, 1999.

[24] M. Keith, "From polychords to polya: adventures in musical combinatorics," 1991.

[25] G. Toussaint, "Classification and phylogenetic analysis of african ternary rhythm timelines," in *Meeting Alhambra, ISAMA-BRIDGES Conference Proceedings*, 2003, pp. 25–36.

[26] F. Gómez, A. Melvin, D. Rappaport, and G. T. Toussaint, "Mathematical measures of syncopation," in *Renaissance Banff: Mathematics, Music, Art, Culture*, 2005, pp. 73–84.

[27] P. C. Vitz, "Information, run structure and binary pattern complexity," *Perception & Psychophysics*, vol. 3, no. 4, pp. 275–280, 1968.

[28] P. C. Vitz and T. C. Todd, "A coded element model of the perceptual processing of sequential stimuli." *Psychological Review*, vol. 76, no. 5, p. 433, 1969.

[29] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8. Citeseer, 2015, pp. 18–25.

[30] M. Grinberg, *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[31] C. Tiefenau, E. von Zezschwitz, M. Häring, K. Krombholz, and M. Smith, "A usability evaluation of let's encrypt and certbot: usable security done right," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1971–1988.

DELIA: A NOISE-BASED VIRTUAL SYNTHESIZER

Cecilia Rossi
Laboratory of Music Informatics (LIM)
Department of Computer Science
University of Milan
cecilia.rossi4@studenti.unimi.it

Giorgio Presti, Federico Avanzini
Laboratory of Music Informatics (LIM)
Department of Computer Science
University of Milan
{name.surname}@unimi.it

ABSTRACT

This paper presents the prototype of a new virtual synthesizer characterized by the idea of generating pitched and non-pitched sounds starting from white noise only, and manipulating it in different ways, such as with subtractive and waveshaping techniques. The possible outcomes cover a palette of intentionally "low quality" sounds and textures, that are meant to be used in sound design scenarios, acusmatic compositions, or simply as original elements in pop songs. The synthesizer is implemented as a VST instrument plugin using the JUCE C++ framework.

1. INTRODUCTION

Luigi Russolo, a member of the Futurist movement of the XX century, in his manifesto "L'arte dei rumori" [1] wrote:

> *Noise is not only loud and unpleasant, you can count a number of noises that are delicate, and provide pleasant acoustic sensations to the ear.*

and further:

> *We need the musicians to get rid of easy and traditional rhythms, they must find in noise a way to grow, to renew, since every noise suggests a blend of the most diverse rhythms.*

It is clear that Russolo believes that noise, in all its facets, possesses its own musical identity and it is surely evident how noise can be either a component or the leading element of Futurist music, which you can interpret as music of the future. In his manifesto he expresses his will to create an orchestra composed of six "noise families" among which it is important to recall the following:

- rumbles, blasts;
- whistles and hisses;
- mumbles and gurglings;
- screechings, creakings, rustlings, buzzes, crackles, creases;

Among his experiments, whose objective was to create an actual instrument that could be part of the orchestra defined above [2], it is important to mention the "Scoppiatore" (the Burster), Russolo's first "Intonarumori" (noise intoner), which emulated a typical bursting noise, and it could change tone in a range of 2 octaves.

Related with Russolo's Intonarumori avant-garde idea, stands the Delia Noise Synth, whose goal is the same: to tune noise. In other words the aim of the project is to make noise "playable" using conventional music practices (i.e., using a keyboard and contemporary music harmony), as well as to produce the aforementioned families of noises.

The name "Delia" was chosen after Delia Derbyshire [3], one of the first electronic music female composers, who was born in Coventry in 1937 and was operational in the BBC Radiophonic Workshop between 1962 and 1973. [1] There she composed, besides the famous "Doctor Who" theme, other extremely creative and experimental works at that time such as "Inventions for Radio" (1964) with playwright Barry Bermange. However the most significant work of hers, which inspired the Delia Noise Synth, is the album "An Electric Storm" (in particular the song "Love Without Sound") realised in 1969 by the White Noise band, founded by Derbyshire, David Vorhaus and Brian Hodgson in 1968. This is a remarkable avant-garde research and accurate experimentation on noise as a full-fledged musical entity.

Among modern days artists, it is important to mention Trent Reznor, the frontman of the American band Nine Inch Nails, who famously said in an interview [4]:

> *I think there's something strangely musical about noise,*

thus reaffirming Russolo's view that noise has impressive musical potential so that it can be seen as a "manifestation of truth" [5, Benjamin Thigpen, p.3].

The purpose of Delia is to make noise more accessible for composers, sound designers or artists in general, in order to let them use it in their compositions, and be able to express all the nuances of their artistic inclinations. Starting from white noise, commonly known as an equal amplitude signal at all frequencies, the user is free to manipulate it: i) through different stages of filtering; ii) through resampling and requantization; iii) through distortion by adding odd harmonics or both even and odd harmonics; not to mention the ability to use noise as it is, in its "natural" shape. All of these possibilities will be discussed in the paper.

Copyright: © 2022 Cecilia Rossi et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[1] https://www.bbc.com/historyofthebbc/100-voices/pioneering-women/women-of-the-workshop/delia-derbyshire, accessed: Feb 08, 2022

2. NOISE AND SYNTHESIZERS

The use of noise was central to the development of experimental music and avant-garde music throughout the 20th century [6], starting with the innovations brought by Edgar Varése in his compositional work and the already cited experiments by Luigi Russolo. In particular, noise has been used as a basic sound source in electronic music since the 1950's, beginning with the explorations carried out by the laboratories in Cologne, Paris, and Milan. Two main scenarios were considered: the manipulation of noises coming from real world recordings, and the electronic generation of white noise to be processed, typically with graphic equalizers [7].

Both scenarios made use of noise in compliance with what has been described in Sec. 1, but due to the technology available at the time the composers could not rely on one comprehensive tool, and instead used a combination of tools to achieve specific goals (i.e. noise sources, equalizers, tape recorders etc.).

When modern normalized synthesizers were introduced to the market, a noise source was almost invariably included, as a companion of one or more periodic oscillators: in most cases the oscillators were meant to produce the main tone, while the noise source was meant to add a stochastic component, aimed at producing a natural sounding output.

This configuration discouraged the use of noise as the main and only source of sound, but at the same time, artists (such as Delia, but also many others such as the krautrock scene or the emerging industrial scene) were looking for tools to introduce noise in pop music [5]. They had to rely on modular tools that, even if designed with this idea in mind, were based on manipulating pitched signals in order to create barely pitched noises (e.g. Buchla, [2] Makenoise, [3] or the "circuit bending" community [8, 9]).

Unlike the above, Delia aims at flipping this perspective: let the user start from noise to design sounds, as in the pioneering times. This approach follows a recent trend of re-discovering through digital technologies the use of noise in 20th century avant-garde music, such as in the digital reconstruction of the Intonarumori family by Serafin and De Götzen [10].

3. DELIA

3.1 Architecture

Delia Noise Synth is a Virtual Studio Technology (VST) [4] plugin developed in C++, using the JUCE [5] framework.

Delia Noise Synth is a polyphonic synthesizer where, as shown in Fig. 1, every voice reflects the traditional structure of a subtractive synthesizer. It is composed of two oscillators, a mixer, a main filter, and an amplification stage. The frequencies of the oscillators and the cutoff frequency of the main filter are controlled by three instances of a *Frequency Control* class, aimed at summing logarithmic frequencies values coming from the keyboard and possible modulation sources, according to corresponding "mod amount" parameters. The mixer, in addition to summing oscillators outputs, can also add to the output the ring modulation between the two oscillators, and additionally high-passes the output in order to remove the DC offset that may be introduced by the ring modulation or the oscillators themselves.

Control signal generators include two Low Frequency Oscillators (*LFOs*), linked to the frequencies of both oscillators, and to the cutoff frequency of the filter, respectively. They also include two envelope generators (*EGs*), linked to the frequencies of the oscillators and filter, and to the amplifier stage, respectively. After all the voices are summed, the signal is passed through a last saturation stage.

Within this general structure, the main distinguishing features of Delia are found in the implementation of the oscillators. The Oscillator class defines the internal structure of the two oscillators and is composed of 4 modules (see Fig. 2), described next.

3.2 Generator

This module deals with source selection and a first filtering stage. You can choose between White Noise or External. The latter refers to the synthesizer external input signal for the first oscillator, while it refers to the first oscillator signal for the second one. In this way the first oscillator can be used to process external signals, while the second one can be used to process the output of of the first one.

An Resistor-Capacitor (RC) filter, characterized by a very moderate slope of 6 dB per octave, is then applied, with a given cutoff frequency determined by the note played and a *tuned* parameter:

$$f_{cut} = 20000 \cdot (1 - tuned) + f \cdot (tuned)\,, \qquad (1)$$

where the *tuned* parameter could vary between 0 and 1 and consequently f_{cut} varies linearly between 20 kHz (for $tuned = 0$) to the frequency f coming from the Frequency Control (for $tuned = 1$).

3.3 Degradation

This module is responsible for resampling the signal from the previous module and re-quantizing it. Signal resampling can be performed in two different ways: by choosing a fixed time in milliseconds, or by using the frequency from the Frequency Control as resampling frequency. By using the latter mode, in conjunction with the previous filtering, a behaviour similar to a comb filter can be obtained, in tune with the note played on the keyboard, as can be heard in the audio samples described in Section 4. Both resampling modes start their clock on a note-on event.

Quantization lets the user choose the number of bits used to represent the signal, allowing to limit the number of possible represented values, introducing sonic possibilities discussed in Sec. 4.

[2] https://buchla.com/

[3] https://www.makenoisemusic.com/

[4] https://developer.steinberg.help/display/VST/VST+Home

[5] https://juce.com

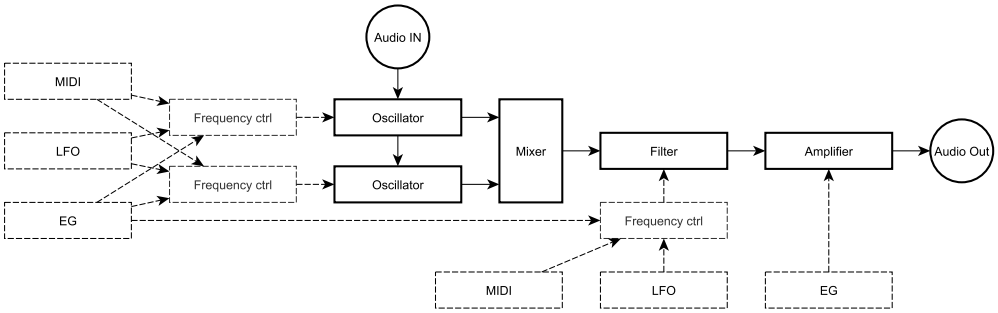


Figure 1. Schematic overview of a single voice of the Delia Noise Synth; Solid elements denotes audio processors and signals, while dashed elements refer to control signal modules and connections.

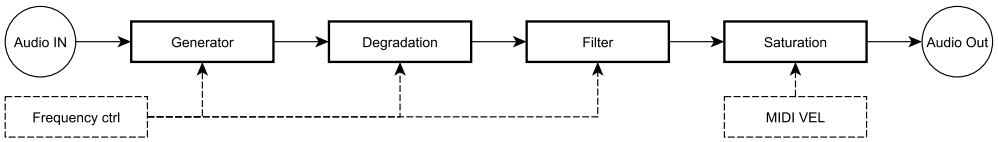


Figure 2. Schematic overview of a single oscillator section; Solid elements denote audio processors and signals, while dashed elements refer to control signal modules and connections.

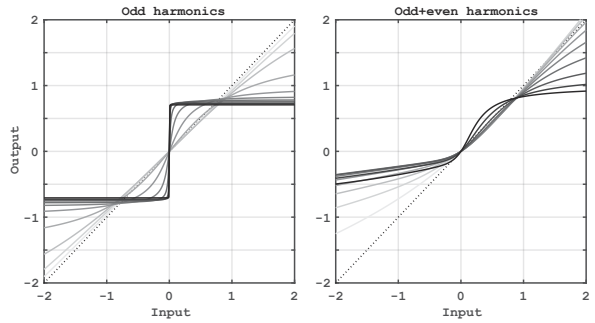


Figure 3. Static response of the saturators used. Different line shades denotes different saturation amount, i.e. the c parameter varying between 0 and 1.

3.4 Filter

This module applies a second filtering stage, which is far sharper than the first one. The filter used is a Topology Preserving Transform filter [11], which which can be considered as a generalization of the bilinear transform for digital filter design [12]. A slope of 24 dB per octave is used, and the user can choose the filtering type (lowpass, band pass, high pass, or none), as well as its resonance, whereas the frequency is controlled by the Frequency Control. This filter is responsible for the oscillator tuning in a conventional sense. The same class is also used in the main Filter section (see Fig. 1), but in that case the cutoff frequency and keyboard tracking are adjustable by the user.

3.5 Saturation

The saturation module applies a distortion to the signal, and it realizes a waveshaping component to create richer waveforms compared to the almost sinusoidal signal that may come out of the filter. Two different types of distortion are implemented.

Odd Harmonics Addition

The distortion function $f_{odd}(x, c)$ used to introduce odd harmonics (typical of square or triangular waves) consists of a simple arc tangent, and it is defined as:

$$f_{odd}(x, c) := \arctan(g_{in}(c) \cdot x) \cdot g_{out}(c) \,, \qquad (2)$$

where x is the input signal, c is the saturation parameter, varying between 0 and 1 along a pseudo-logarithmic scale, $g_{in}(c)$ and $g_{out}(c)$ are a distortion factor and a gain compensation factor, respectively.

The resulting distortion function is visible in Fig. 3 (left).

Odd and Even Harmonics Addition

A distortion function $f_{even}(x, c)$ adds odd harmonics as well as even harmonics in the signal (typical of sawtooth waves). This is performed through the soft++ function [13] $s_{++}(x, c)$, followed by an arc tangent function identical to the one used for odd harmonics. The soft++ function is a simplified version of the one described in [13] and is defined as:

$$s_{++}(x, c) = \ln(1 + e^x) + (x/q(c)) - \ln(2) \,, \qquad (3)$$

where x is the input signal, c is the saturation parameter, $q(c) := 8c+2$ is the distortion amount, and $\ln(2)$ is an offset value which ensures that the distortion function passes through zero, thus avoiding an offset when no sound is produced.

The $f_{even}(x, c)$ distortion is defined as:

$$f_{even}(x, c) := f_{odd}(\, s_{++}(g(c)x, c)/g(c),\ \ 0.5c) \,, \qquad (4)$$

where $g(c) := 10c + \epsilon$ is a gain compensation factor used to balance the effect of the two saturation functions.

The resulting distortion function is visible in Fig. 3 (right).

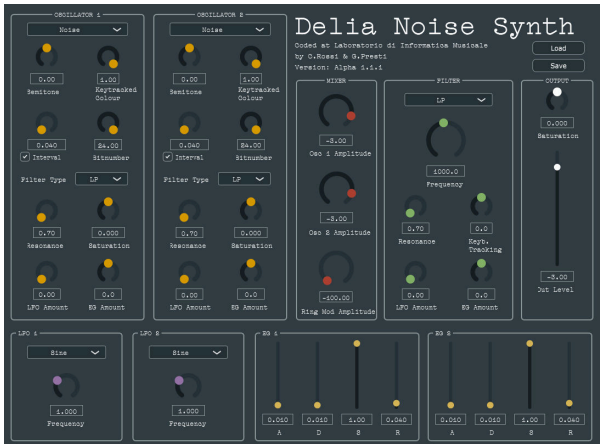


Figure 4. The graphical user interface of Delia Noise Synth.

4. DISCUSSION

The VST3 build of Delia Noise Synth can be downloaded from a dedicated webpage and may be tested with any VST host. [6] A picture of the graphical user interface, exposing all the parameters discussed in the previous section, is visible in Fig. 4.

Sound examples of the output obtainable with Delia are also available at the same dedicated webpage. Except for the final music sample, all the examples discussed in the remainder of this section were produced using the oscillators only (since the effects of the rest of the sound processing chain are consistent with standard synthesizer use practices), in order to demonstrate the sonic possibilities of the synthesizer.

4.1 One Oscillator

Here the effects of progressively including the oscillator components of Fig. 2 in an incremental fashion are discussed. The first examples are a demonstration of how Delia oscillators can behave like traditional oscillators (with the difference of being "imperfect", which is the point of the project). The subsequent examples instead demonstrate some of the peculiar features of Delia.

Generator

By filtering the source noise with a narrow band-pass filter, it is possible to generate an almost-sinusoidal sound. The randomness of the source noise provides to the resulting signal with noticeable imperfections and "movement". Further stages of processing (via second oscillator or the main filter) may result in a more precise signal.

Waveshaping

Adding the odd-harmonics waveshaper to the processing chain results in an almost-square wave, inheriting the same imperfections of the sinusoidal signal. Note that extreme saturation values will result in a naive-square wave, aliasing could have been avoided by oversampling the signal, but we left this choice to the user just in case aliasing is a desired feature.

Similar considerations apply when the $f_{even}(x, c)$ distortion function is used.

Quantization

As an effect of quantizing a noise signal, a rich non-pitched texture with an interesting low-end is produced.

Downsampling

As an effect of downsampling noise, with the sampling frequency being controlled by the played note, a peculiar intonation effect is obtained (the online example demonstrate this through a sequence of three notes).

Downsampling and Filtering

In this case the noise is downsampled to a very low rate (in the order of some tenth of a second) so to produce a rhythmical sequence of steps, a strategy used in Low Frequency Oscillators (LFO) usually referred as Sample & Hold (S&H). After the downsampling stage, a severe quantization reduces the possible step sizes, merging steps with similar amplitude values, resulting in an irregular pattern of steps. Finally, the oscillator filter introduces a very high resonance on the played note, turning each remaining step into a tuned impulse. In the audio sample a major chord is played, but due to the oscillator setup, it sounds like a random arpeggio.

4.2 Two Oscillators

Whereas the examples discussed above use a single oscillator, the following focus on the combination (in series or in parallel) of the two oscillators.

Series Connection With Quantization

Quantization noise introduced by the second oscillator can be used to enrich the base waveform generated by the first one, creating a variety of textures.

Series Connection With Downsampling

The aliasing introduced by downsampling the second oscillator can also be used as a way to enrich the signal produced by the first one, resulting in different pad sounds.

Parallel Connection

Using the two oscillators in a parallel fashion allows to exploit each of them for different musical purposes. As an example (see the related online sample in the dedicated webpage), the first one may be used to produce rhythmic patterns, while the second one produces continuous pad sounds.

Ring Modulation

The ring modulation capabilities of the mixer stage can be used creatively to further enhance the palette of sonic outputs. As an example (see the related online sample in the dedicated webpage), one oscillator can be used as a S&H LFO, which multiplies a continuous pad sound produced by the second oscillator.

[6] https://www.lim.di.unimi.it/demo/delia.php.

A Musical Example

As a final demonstration of Delia, a music sample was produced by exploiting the aforementioned techniques together with other Delia features (i.e., the EGs and the main filter, see Fig. 1). The output from Delia was only processed through a reverb effect only, and no additional equalization nor compression were applied.

5. CONCLUSIONS

A new virtual synthesizer called Delia Noise Synth was presented in this paper, and its capabilities were exemplified through a number of examples. The main distinguishing features of Delia are found in the definition of its oscillators, aimed at letting the user start from noise to produce a variety of sounds, in contrast with traditional oscillators which generate quasi-periodic, spectrally-rich waveforms as raw material for subtractive synthesis. It is our hope that the change of perspective implied by these features can help the creative process followed by the user to reach a desired effect.

6. REFERENCES

[1] L. Russolo, *L'arte dei rumori*. Edizioni futuriste di "poesia", Corso Venezia, 61 - Milano, 1916.

[2] B. Brown, "The noise instruments of Luigi Russolo," *Perspectives of New Music*, vol. 20, no. 1-2, pp. 31–48, 1981.

[3] T. Winter, "Delia Derbyshire: Sound and Music for the BBC Radiophonic Workshop, 1962-1973," Ph.D. dissertation, University of York, 2015.

[4] G. Rule, "Trent Reznor," *Keyboard Magazine*, March 1994, accessed: Feb 08, 2022. [Online]. Available: https://www.theninhotline.com/archives/articles/manager/display_article.php?id=548

[5] A. Cassidy and A. Einbond, Eds., *Noise in and as Music*. University of Huddersfield Press, 2013.

[6] P. Hegarty, *Noise/Music: A History*. New York: Continuum, 2007.

[7] M. M. Novati and J. Dack, *The Studio Di Fonologia: a musical journey 1954-1983, update 2008-2012*. Ricordi, 2012.

[8] R. Ghazala, *Circuit-Bending: Build your own alien instruments*. John Wiley and Sons, 2005, vol. 15.

[9] G. Hertz and J. Parikka, "Zombie media: Circuit bending media archaeology into an art method," *Leonardo*, vol. 45, no. 5, pp. 424–430, 2012.

[10] S. Serafin and A. De Götzen, "An enactive approach to the preservation of musical instruments – Reconstructing Russolo's Intonarumori," *J. of New Music Research*, vol. 38, no. 3, pp. 231–239, 2009.

[11] V. Zavalishin, "The art of va filter design," *Native Instruments, Berlin, Germany*, 2012.

[12] U. Zölzer, *Digital audio signal processing*, 2nd ed. John Wiley & Sons, 2008.

[13] A. Ciuparu, A. Nagy-Dăbâcan, and R. C. Mureşan, "Soft++, a multi-parametric non-saturating non-linearity that improves convergence in deep neural architectures," *Neurocomputing*, vol. 384, pp. 376–388, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219317163

FOR LISA. MUSIC COMPOSITION FROM GRAVITATIONAL WAVES

Andrea Valle
CIRMA/StudiUm - Università di Torino
andrea.valle@unito.it

Valeriya Korol
Institute for Gravitational Wave Astronomy
& School of Physics and Astronomy,
University of Birmingham
korol@star.sr.bham.ac.uk

ABSTRACT

In the paper we discuss the composition of a piece for prepared piano, *Periplo del latte*, that takes into account as its starting point the simulated gravitational wave data for the future LISA space mission. First, we introduce the LISA project and its output. Then, we present *Einstein's Sonata*, a multimedia project devoted to the artistic public display of the LISA data. *Einstein's Sonata* features as its main element the piece *Periplo del latte*. The latter is based on mapping astronomical data onto music. We thus detail a four-stage procedure for algorithmic composition, that features two data preprocessing stages, a mapping into an abstract control space and finally automatic notation generation.

1. INTRODUCTION

Recently astronomers started exploring the Universe through a new kind of radiation: gravitational wave radiation or gravitational waves. Gravitational waves represent perturbations of the space-time generated by masses in acceleration. Their properties are similar to those of electromagnetic waves: they are traversal waves that propagate at the speed of light and can be characterized by properties like frequency and amplitude. Unlike electromagnetic waves that can be seen as visible light or felt as heat, gravitational waves have been conceived theoretically and cannot be perceived directly. Thus, music rendering of gravitational waves provides a way of experiencing and interpreting these otherwise imperceptible waves.

In this work we consider gravitational waves detectable with the future *Laser Interferometer Space Antenna* (LISA), an European Space Agency-led mission. Consisting of three identical spacecrafts in an equilateral triangle configuration distant 2.5 Mkm apart and connected by laser links, LISA will be sensitive to a part of gravitational wave spectrum between $10^{-4} - 10^{-1}$ Hz [1]. A large variety of astrophysical sources emit gravitational waves in this frequency band ranging from stellar remnants (black holes, white dwarf and neutron stars) in binary systems in our Milky Way to nascent massive black holes early in the history of the Universe [2]. However, gravitational wave signals from binaries composed of two white dwarf stars – hereafter double white dwarf (DWD) – will outnumber all the other astrophysical gravitational wave sources in the LISA frequency band by at least an order of magnitude, and, thus, are the focus of our work. As many as tens of millions of DWDs are expected to emit gravitational waves in the LISA band, out of which tens of thousand will be detected and characterized by LISA [3].

The estimated number of LISA detectable DWDs is based on theoretical studies of these binaries in the Milky Way realized in preparation for the LISA mission by combining many numerical techniques. The first is to model the intrinsic properties of DWDs – binary orbital geometry and white dwarf masses that determine the strengths of its gravitational wave signal – with a technique called binary population synthesis [4]. It represents a collection of numerical prescriptions (typically motivated by available electromagnetic observations) for processes involved in stellar and binary evolution. These prescriptions are combined in a numerical code that "evolves" a binary system from the birth of two stars until both turn into white dwarfs [5, 6]. Second, a detailed Milky Way model describing its shape is used to realise a mock galaxy [7]. Practically, it serves to assign 3D positions to synthetic DWDs with respect to LISA that also affects the strength of the gravitational wave signal, which is inversely proportional to the DWD distance from the LISA detector. Next, knowing the position and properties of DWDs, their gravitational wave signals can be fully modelled and the detectability of binary can be assessed by processing them with the LISA detection pipeline, which simulates LISA's response to the incoming gravitational wave [8]. Lastly, detectable DWDs are collected in a catalogue listing properties for each binary (Section 2). A similar catalogue can be expected as part of data products generated by the LISA mission. We illustrate the spatial distribution of DWD detectable by LISA in Figure 1.

As communication of scientific results to a general audience is becoming increasingly important, we developed the multimedia project, *Einstein's Sonata*, with the aim of providing an artistic rendition that could express in a creative and accessible way (a part of) the output of the LISA mission. *Einstein's Sonata* was born from the idea of transforming theoretical models of gravitational waves signals emitted by DWDs populating our Milky Way into musical and visual arts. Based on the concept of music being a powerful tool of engagement, the goal of this project is to create music by transforming astronomical

Copyright: © 2022 Andrea Valle et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

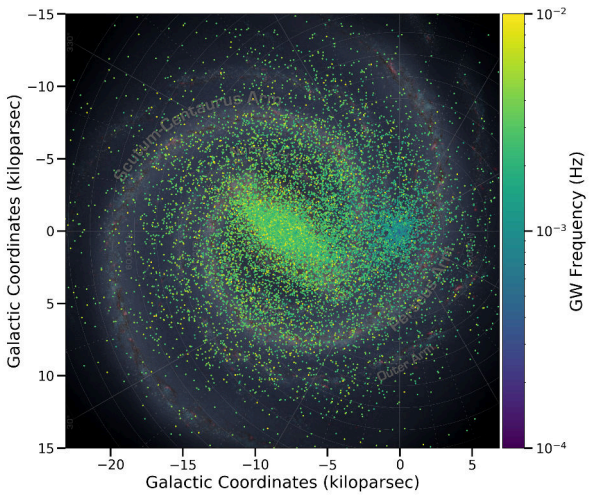


Figure 1. Spatial distribution of synthetic DWD binaries detectable with LISA. In figure DWD are plotted using galactic coordinates centered on the Sun such that the synthetic Galaxy is seen from above and the LISA detector is located approximately at $(0, 0)$. In addition, DWDs are color-coded by their gravitational wave frequency. In the background the artist impression of our current view of the Milky Way. To construct this figure we used DWD simulation of [9].

data into sounds and to contribute to the cultural heritage of the community focusing on both sciences and arts. The visual part of *Einstein's Sonata*, conceived by the artist Samantha Stella, is inspired by the opening scene of the film *Drowning by Numbers* (1988) by British film director and artist Peter Greenaway. The film opens with a little girl - adorned in a dress from Spanish Baroque painter Diego Velazquez's *Las Meninas*, 1656 – rope-skipping while counting stars from 1 to 100. The performance of the *Sonata* opens with Samantha Stella reading the names of the 100 brightest stars in the Milky Way symbolising future "brightest" gravitational waves sources that will be discovered by LISA. The performance acts as a framework around the key element, a piece for prepared piano entitled *Periplo del latte* composed following a musification strategy that takes into account the experimental data. The title of the project, *Einstein's Sonata*, represents a homage to Albert Einstein, who postulated the existence of gravitational waves in the framework of the theory of General Relativity dated 1916 [10]. However, the (direct) detection of gravitational waves became possible almost 100 years later, when Laser Interferometer Gravitational Wave Observatory (LIGO), a ground-based gravitational wave detector, caught a signal generated by the pair of colliding black holes nearly 1.3 billion light years away [11].
In the following, we describe the musification strategy that was used to compose *Periplo del latte*, the piece for prepared piano. Again, the title ("periplus of the milk") is a reference in Italian to the exploration of the space of the Milky Way.

Quantity	Units
frequency f	Hz
frequency derivative $\dot{f}$	Hz/s
amplitude $\mathcal{A}$	dimensionless
chirp mass $\mathcal{M}$	$M_\odot$
distance d	kpc
sky coordinates (θ, ϕ)	radiants
signal-to-noise ratio SNR	dimensionless

Table 1. A table of DWD parameters listed in the input data file used for musification.

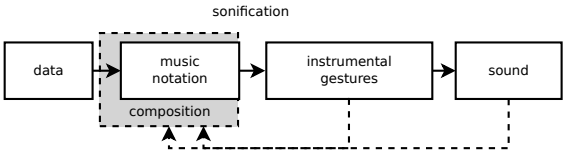


Figure 2. Sonification and music composition.

2. DATA AND FORMAT

Gravitational wave signals emitted by DWDs can be considered as quasi-monochromatic and can be fully described by 8 parameters: amplitude, frequency, frequency derivative (change of the signal's frequency with time), sky coordinates in ecliptic coordinate system (i.e. a spherical coordinate system centred on the position of the Sun and aligned with the ecliptic), binary orbital inclination, gravitational wave polarisation and initial orbital phase. The detectability of each DWD signal is assessed based on the signal-to-noise ratio (SNR), which expresses the strength of the gravitational wave signal with respect to the LISA instrumental noise and can be interpreted as a measure of relevance. In the context of the LISA mission, SNR $= 7$ is typically required to detect a monochromatic source, while a higher threshold guarantees the measurement of all eight parameters describing the signal in addition to the detection. The chosen threshold yields a catalogue of $\sim$ 26k DWDs.

For music purposes we consider LISA detectable DWDs (i.e. those with SNR > 7) only and use parameters listed in Table 1. Note that this list ignores some of the astrophysically less interesting parameters and explicitly specifies binary's chirp mass – a combination of the binary components masses that drives frequency evolution of the signal – and distance that can be derived from the measurement of the amplitude and frequency derivative [12].

To summarize:

- the input data represent a set of measurements (in a model) in relation to points in a space;
- in terms of dimension, the catalog – even if already resulting from a selection based on SNR – contains more than 26k measurements, each provided with its 8 parameters.

While there were no strict constraints on the commission side, it was evident that a methodologically-wise design was required in order to respect the scientific nature of the project while composing the music. Thus, we defined a sonification-inspired composition procedure. The sonification of astronomical data has a long tradition [13–15]. An up-do-date discussion with references to previous cases is provided by [15] in the context of their sonification of zCOSMOS, an astronomical dataset that contains information about $\sim$ 20k galaxies. In all these projects the typical output is MIDI or electronic sound, as it is easier to define various mapping strategies within a completely digital environment. Differently from these examples, we decided to compose for piano, so to propose the result in an acoustic concert setting and as a part of the physical setting of *Einstein's Sonata*.

Instrumental music composition can be thought of as a special case of sound design to be realized by mapping abstract sonic features onto instructions for musicians, the latter acting properly as acoustic sources. In this framework, music can be thought of as a sonification of gestures [16]. Instrumental gestures cannot be addressed directly, rather they require the mediation of music notation. Thus, in the case of music composition for acoustic instruments, a sonification-based procedure must result into graphic symbols, rather than ending with audio signals. To sum up, far from being immediate, sonification for instrumental composition implies a chain of transformation processes (Figure 2). A composition work (dashed lines) that is exclusively oriented towards acoustic instruments properly deals with the process involving the first two domains (data and music notation), the second being its output. Instrumental gestures and sound, and the process linking them, have to be taken into account as possible constraints while composing, in a feedback loop. Data-driven approaches to composition, that deal both with utilitarian and creative ends and constraints, are referred to as "musification" [17]. In the following sections, we will thus describe a musification procedure that yields to a music composition based on gravitational data. The process is properly a musification rather than a sonification because – as we will discuss– data have been largely simplified in order to obtain a piano piece (see later). Assuming the perspective of data compression, such a simplification can be thought of as "lossy" (vs. "lossless"): some data are discarded in an irreversible way, yet the content is mostly preserved. In our musification strategy, simplified gravitational data are still the source of all music events.

3. A FOUR-STAGE MUSIFICATION SCHEME

In relation to the previously discussed sonification chain, we thus designed a pipeline linking data to music notation (Figure 3). In this parameter-mapping approach, gravitational wave data (GWD) transit through a four-stage architecture so to end into music notation. Notation acts as a further symbolic, graphical, layer yet to be aurally realized by the piano player, finally resulting into sound. The four stages can be grouped into two subsets. Stages I and II map GWD into an abstract space model, while stages III and IV take the output of II in order to automatically generate music notation. Each subset features as its first stage a Preprocessor that filters and maps data so that they can be used in the subsequent one. The ratio for splitting the *data* $\rightarrow$ *notation* process into two subsets lies in the spatial nature of data. Thus, in stages I and II we first construct an abstract space that can be explored. Then, after obtaining a set of time series from such explorations, we map them into music notation (stages III and IV). In short, stages I and II are related to "space modeling" while III and IV deal with "music interpretation" (see Figures 4 and 9, discussed later). The final result is a piece made up of 11 short sections representing into notation different explorations of the data space.

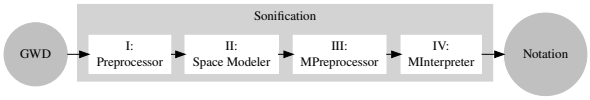


Figure 3. Four-stage sonification architecture.

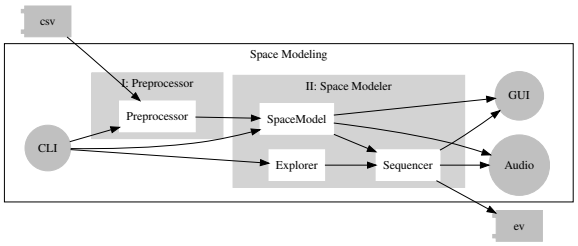


Figure 4. Stages I and II: Space modeling.

4. SPACE MODELING

Stages I and II deal with space modeling. The aim of space modeling is to obtain a simplified 2D model that can be explored interactively so to gain information to feed composition. Space modeling (stages I and II) is entirely implemented in the SuperCollider interpreted language [18], that provides facilities both for data manipulation and audio/visual displaying. A detailed description is given in Figure 4, to which we refer in the following.

4.1 Stage I: Preprocessing

GWD are passed to the Preprocessor via a file in CSV format. The main issue in the musification design is the large amount of data to be taken care of. This is particularly relevant if we consider that the final output is music notation, to be passed to the piano player. The graphic output cannot be simply a graphical data visualization, rather it is constrained by a maximum complexity in relation to music practice. This aspect requires that the so-called Common Practice Notation (even if extended, see later) is taken into account. A filtering process is thus required to perform data reduction. Once the CSV file has been parsed, the Preprocessor operates in three steps:

- the SNR parameter is used again to filter out more data. With SNR$=$ 125, 843 (over $\sim$ 26k) sources

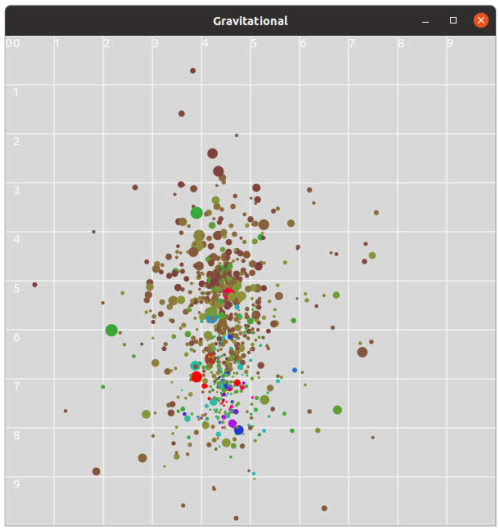


Figure 5. SuperCollider 2D display for the SNR filtered sources. Each circle represents a sound/gravitational wave source with a radius proportional to the source frequency. The color coding reflects the source amplitude: the amplitude increases from brown/green, to blue/violed to red.

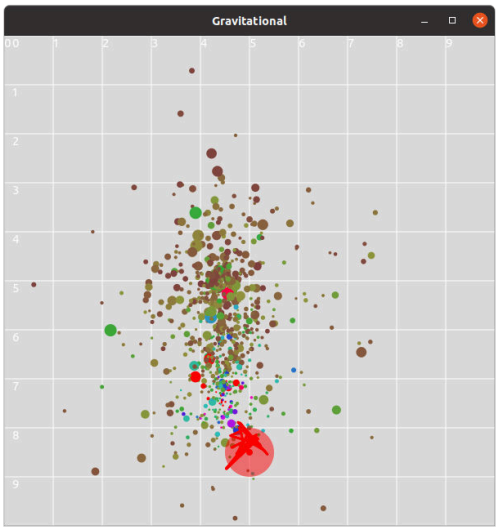


Figure 6. Trajectory (red broken line) and audibility radius (red circle) in the SuperCollider application.

remain available, the most relevant ones.

- spatial dimensions are reduced to 2 by keeping distance and ϕ while discarding θ. This results in a 2D projection of the original space.
- apart from spatial coordinates, frequency and GW amplitude are the sole parameters that are kept, while the other are discarded.

Finally, the Preprocessor normalizes 2D sources position in a $[0.0, 1.0]$ range for sake of simplicity, so that SNR-filtered sources occupy the whole normalized space. The Preprocessor can be tuned interactively by coding in SuperCollider.

4.2 Stage II: Space modeler

Data reduction performed on source data by the Preprocessor allows to obtain a 2D visual display useful for a compact exploration of the overall data space. While visual displaying is independent from sonification (which can be thought of as a form of aural displaying), if the two strategies are consistent it becomes easier to explore the data set, in order to understand its features. Figure 5 shows the 2D space in the interactive SuperCollider GUI. Each source is represented by a dot circle with the radius proportional to frequency. The color of each circle is obtained by using a hue-saturation-value color model, in which the three dimensions are scaled proportionally to the source amplitude. This can be read as: the higher the amplitude, the more saturated and brighter the color. In terms of hue, amplitude increases from brown/green to blue/violet to red. Again, interactive control on visualization is made possible by coding in the SuperCollider environment.

In this spatial model, the musification metaphor is based on considering each GW source (i.e. circle in the GUI) as a sound emitting source. Each of these sound sources is provided with two features: a characteristic frequency and an emission frequency. Characteristic frequency indicates a specific pitch while emission frequency is related to a repetition rate. In short, each source is given a "pitch" to be repeated at a certain rate. The resulting acoustic space is thus meant as a set of pulsating, pitched sources. In terms of parameter mapping, the GW frequency is mapped onto the emission frequency (rate), while the GW amplitude is inversely mapped onto characteristic frequency (pitch), so that highest amplitudes result into lowest pitches. Such an arrangement directly derives from the observation of the organization of the sources in space. Sparse sources at the periphery of the space will typically have low GW amplitude yielding high pitches. The center of the space is dense and includes most of sources with highest amplitudes. The musification of this region will consequently feature the lowest notes (together with high ones). This abstract acoustic space is meant to be navigated by an Explorer. The main metaphor is travelling through the space and detecting sources while approaching them. The Explorer is defined by a trajectory, to be traversed at constant speed, and by an audibility radius a_r. Figure 6 shows a trajectory (red line) with a_r for its starting point (red circle), showing the audible sources for that position.

Only sources inside a_r are taken into account, while the ones lying outside are discarded. The spatial metaphor is perceptually viable: sources near the Explorer are audible, while sources beyond a_r cannot be heard. The Explorer is able to detect a source within its audibility radius and measure its distance. Such a mechanism has some interesting features on the composition side:

- trajectories introduce a time-based behavior in the space;
- trajectories set the pace of the music (though the control of their length and speed);
- trajectories select which part of and how the space is to be heard, e.g. they can be tuned by observing the

visual display;
- a_r represent a further filtering over sources;
- distance between Explorer and each source provides a viable, ecological parameter to describe sonic transformations.

The trajectory/audibility radius framework has been introduced by [19] in relation to dynamic soundscape simulation. Such an ecological metaphor [15] provides a link between source data and auditory perception and places the composition technique, even if indirectly, in the context of "soundscape composition" [20]. To sum up, GW sources are thought as elements contributing to an abstract but motivated soundscape that changes dynamically as a listener moves along an exploring path. In the space model, trajectories allow to explore different parts of the space while variably configuring the available parameters, e.g. a_r can be set in relation to each trajectory. Thus, the final piece *Periplo del latte* is a cycle made up of shorter sections, each one implementing an exploration along a specific trajectory. Various algorithms have been designed to construct trajectories. As an example, the trajectory in Figure 6 results from a Brownian motion constrained by a density parameter, so that new points always bounce inside a certain region. This feature makes it possible to explore a limited set of sources in a dynamical manner.

With reference to Figure 4, the Sequencer is passed the SpaceModel and the trajectory created by the Explorer. The Sequencer generates sequences of events for sources inside a_r while moving along the trajectory over time. For each active event (which rate depends on the source frequency), the attack time is retained and the GW amplitude is mapped logarithmically in the audio frequencies corresponding to the MIDI range $[21, 108]$ (piano pitches). Also, the distance with the actual point on the trajectory is computed.

The output format for each event is `att, freq, dist` and this information is logged to the `ev` CSV file.

The whole composition process can be fine tuned by visualizing in real time trajectories while simulating the sound output by means of a synthesized piano (GUI and Audio in Figure4). Apart from sound simulation, and even if the output acoustic frequencies are already constrained in the piano range ($[27.5, 4186]Hz$), the final result of stage III is still abstract from sound implementation, and the resulting sequenced events can be used to feed e.g. a digital sound synthesis process, even in real time. In this case, as the software architecture is modular, it is easy to replace a previously computed trajectory with a point input by the user.

5. M(USIC)INTERPRETATION

The main project constraint was to write for piano solo. In order to disconnect piano from its standard music associations and to create a mood of otherness related to such distant and literally invisible objects as gravitational wave sources, the piano has been prepared (Figure 7). Metal chains have been inserted across the string length for the whole string set, so to produce a bright, buzzing (rather than bouncing) timbre that still does not interfere with pitch discrimination. Chain weights are to be accommodated in relation to string diameter, so to ensure a consistent effect all over the strings and to be consistently sensitive to dynamics. Also, paper strips are to be placed under all the dampers. These two preparations shift the piano sound towards a tuned percussion instrument while retaining its typical keyboard agility.

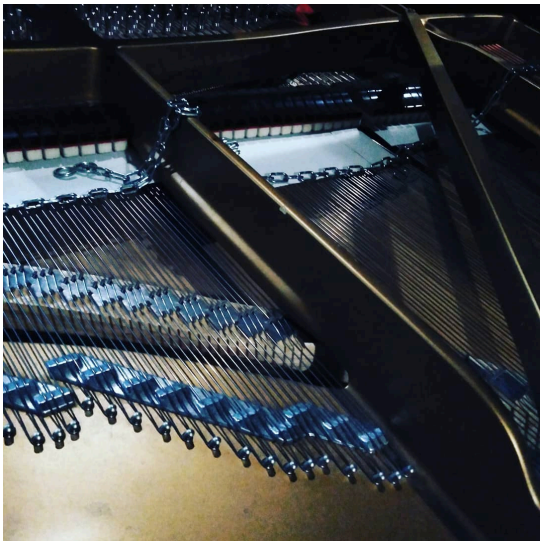


Figure 7. Piano preparation.

In relation to musification, some other constraints cascade from writing for piano (pc), that we will address in the next sections:
- 88 pitches are available in 12-tone equal temperament (pc_1);
- a theoretical maximum of 10 simultaneous notes are available resulting from key pressed by the player. Moreover, hand positions should be addressed as they constraint the available notes that can be played together (pc_2);
- a representation format (notation) must be devised that can compactly express the information related to events while still being playable (pc_3).

Stages III and IV are entirely implemented in Python [1], that is –as SuperCollider– a high level, interpreted language, well fitted for music composition.

5.1 Stage III: MPreprocessor

The Sequencer outputs the `ev` log file which is passed to the M(usic)Preprocessor (Figure 9). For each event, the three associated `att, freq, dist` values are remapped as follows:
- the attack of each event is rounded to the nearest value in relation to three available grids quantizing a beat in $5, 6, 8$ subdivisions and assuming a bpm with $\frac{1}{4} = 60$;
- the frequency is converted into its integer MIDI note value, so that it can be mapped onto a single piano key (see pc_1);

[1] `https://www.python.org/`

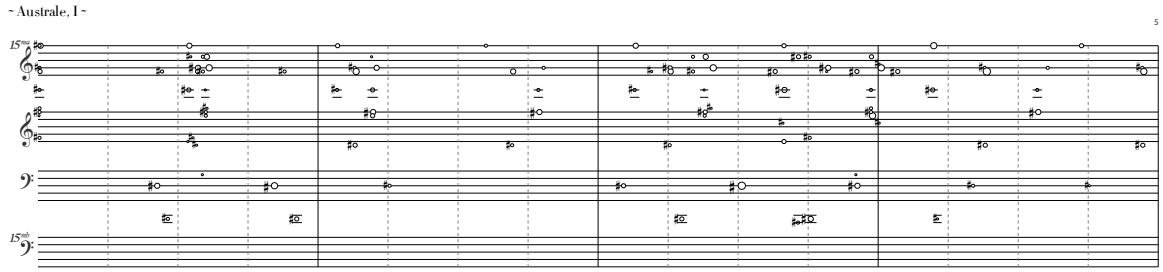


Figure 8. *Australe, I*, first four measures.

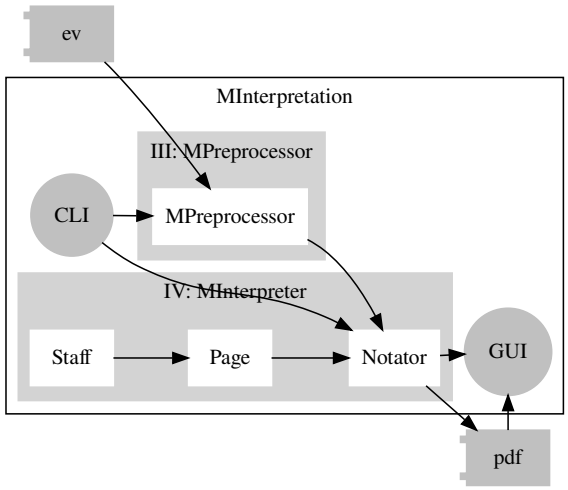


Figure 9. Stages III and IV.

- distance, which range is comprised in $[0, a_r]$, is mapped linearly on an inverted 5-step scale representing the event dynamics, so that minimum distance is mapped onto 5 (= *ff*) and maximum one onto 0 (= *pp*). This is indeed a reference, even if largely "cartoonified", to intensity as a monaural psychoacoustic cue for distance detection in space [21].

In order to comply with pc_3, Common Practice Notation (CPN) has been taken into account as the main reference for the piece. In relation to the former, some issues have emerged. While the music is metrically organized by means of proportions (see before), still parallel irregular grouping are very complex to handle in terms of graphic layout of CPN. Another issue is that large clusters of notes can become cumbersome on a standard piano two-staff system (the so-called "grand staff"). Since the beginning of 20th century, many composers have intensively worked on alternative music notation styles [22], which are now part of the standard background of contemporary musicians and composers [23]. Drawing inspiration from piano notation in Helmut Lachenmann's *Allegro sostenuto* [2], we thus devised a specific notation format based on four staves, in which two staves surround the standard piano grand staff. Figure 8 provides an example by reproducing the first four measures of an actual section of *Periplo*. Here, the upper and the lower staves extend respectively the treble and the bass clefs, and they sound respectively 2 octaves up and down (*15ma*). Two lines are thus required between the two treble clefs and between the two bass clefs in order to ensure the correct pitch progression. Time representation makes use of the so-called "time notation" in which duration is proportional to graphical space in the staff (see [22], chap. 3). In Figure 8, each measure represents four seconds. In short, chronometric time is represented by assuming a meter $= \frac{4}{4}$ and a tempo of $\frac{1}{4}$ = *60* bpm. In the notation, dashed vertical lines represent 1-second duration ranges ($= \frac{1}{4}$) and are provided as a reference for the player. In the performance, tempo can be slowed, provided that it is constant over each section. Notes are represented as white circles. White color is not related to duration as in CPN. In fact, duration is not specified at all in the notation: it must be accommodated by the player in relation to performance and musical needs (including e.g. the use of the pedal). On the contrary, for each note the attack time is notated precisely, and proportionally to space. Each note's attack is represented by the foremost left point on the note's circle symbol. As notation is proportional and not metric, the three proportions $(5, 6, 8)$ are not explicitly notated, rather they represent a sort of hidden layer of time organization emerging in the performance. For each note, the circle diameter indicates its dynamics, from smallest (= *ppp*) to largest (= *fff*). In this way, dynamics progression (crescendo/diminuendo) for repeated notes with the same pitch can be easily detected in the score. Alterations are notated exclusively by means of sharp accidentals (♯). They apply only for the single note for which they are specified. The resulting, pulviscular notation is also meant as an apt reference to stars and constellations.

5.2 Stage IV: MInterpreter

Such a custom notation format prevents the usage of available packages for automatic music notation generation ([24–27], or the Python-based [28]). Rather, a specific implementation has been developed, integrating SuperCollider with Python in a "fluid" architecture [29]. As shown in Figure 9, remapped data from MPreprocessor are now suitable for the Notator module that is responsible for notation generation in a completely automatic fashion. Notator handles the generation of the final graphical elements in PDF format by exploiting Shoebot, a Python library for

[2] H. Lachenmann, *Allegro sostenuto*, Breikopf & Härtel, Leipzig, 1986/88, KM 2407.

vector graphics [3] . For each piece, a score is made up of pages, in turn made up of staves, containing custom music symbols. The last two layers are handled by two separate Python classes, Page and Staff (Figure 9). Staff implements the custom extended grand staff discussed before. Notator examines the event log for each section and first defines the required number of Page instances. For each Page, it creates two Staff instances. For each event, att is then used to define a certain position in the correct Page and Staff. The vertical position on the Staff depends on the pitch while the radius of the note circle symbol is proportional to dynamics. Graphic generation is fine-tuned interactively by monitoring the generated PDFs and subsequently modifying Python settings.

Events and their organization over time are not formally defined per se, rather they result from the exploration of the space, thus making very difficult to predict a priori the actual output. The latter is a particularly severe issue in relation to pc_2 (piano realization). Interactive monitoring makes possible a trial-and-error approach to assess the overall density of notes in terms of performing possibility, modifying the input parameters if the score cannot be played (e.g. too many notes or too distant finger positions). As a safeguard, in case of complex blocks of notes, the piano player is explicitly left the decision whether to omit some notes for sake of feasibility.

All the pages resulting from automatic notation generation are then assembled by scripting ConTEXt, a TEX-based package for document preparation [4] . The final score of *Periplo del latte* consists of 11 sections. Figure 8 shows the first four measures (top half-page in landscape format) of the *Australe, I* section. The piece results from the exploration shown in Figure 6.

6. CONCLUSIONS

Our musification project had to address many different issues: from a large amount of data to their spatial internal organization, from instrumental composition to automatic generation of music notation. The modular four-stage architecture has allowed us to cope with such issues so to integrate them into a unified musification model. *Periplo del latte* debuted in the context of *Einstein's Sonata* on October 29th, 2021 at Festival della Scienza, Genoa [5] . The festival is meant to overcome the traditional opposition between scientific and humanistic culture and its intended audience is as general as possible. *Einstein's Sonata* has been preceded by a talk discussing both the original experiment and the musification process. This public display has demonstrated the potential of musification not only as a scientific data display tools, but in our case as a way to bring a general public closer to advanced scientific concepts by means of music while respecting the scientific nature of the project. At the same time, it has allowed us to introduce sonification and musification per se, largely unknown to a wider audience.

The recording of the piece is available at `https://andreavalle.bandcamp.com/album/periplo-del-latte`.

[3] `http://shoebot.net/`

[4] `https://wiki.contextgarden.net/`

[5] `http://www.festivalscienza.it/`. An artistic teaser, alluding to all the components involved in the project, can be found here `https://www.youtube.com/watch?v=q_zZu47ky5I`

Acknowledgments

The authors would like to thank the team of the *Einstein's Sonata* project, Samantha Stella (artistic director), Luca Ieracitano (pianist), and Nicola Tamanini, who contributed to the project at initial stages. This works has been funded by the by the Gruber and Rubicon postdoctoral fellowships (grant number 019.183EN.015) awarded to Valeriya Korol respectively by the International Astronomical Union (IAU) and the Netherlands Research Council (NWO).

7. REFERENCES

[1] P. Amaro-Seoane, H. Audley, S. Babak, J. Baker, E. Barausse, P. Bender, E. Berti, P. Binetruy, M. Born, D. Bortoluzzi, J. Camp, C. Caprini, V. Cardoso, M. Colpi, J. Conklin, N. Cornish, C. Cutler, K. Danzmann, R. Dolesi, L. Ferraioli, V. Ferroni, E. Fitzsimons, J. Gair, L. Gesa Bote, D. Giardini, F. Gibert, C. Grimani, H. Halloin, G. Heinzel, T. Hertog, M. Hewitson, K. Holley-Bockelmann, D. Hollington, M. Hueller, H. Inchauspe, P. Jetzer, N. Karnesis, C. Killow, A. Klein, B. Klipstein, N. Korsakova, S. L. Larson, J. Livas, I. Lloro, N. Man, D. Mance, J. Martino, I. Mateos, K. McKenzie, S. T. McWilliams, C. Miller, G. Mueller, G. Nardini, G. Nelemans, M. Nofrarias, A. Petiteau, P. Pivato, E. Plagnol, E. Porter, J. Reiche, D. Robertson, N. Robertson, E. Rossi, G. Russano, B. Schutz, A. Sesana, D. Shoemaker, J. Slutsky, C. F. Sopuerta, T. Sumner, N. Tamanini, I. Thorpe, M. Troebs, M. Vallisneri, A. Vecchio, D. Vetrugno, S. Vitale, M. Volonteri, G. Wanner, H. Ward, P. Wass, W. Weber, J. Ziemer, and P. Zweifel, "Laser Interferometer Space Antenna," *arXiv e-prints*, p. arXiv:1702.00786, Feb. 2017.

[2] P. Amaro-Seoane, J. Andrews, M. Arca Sedda, A. Askar, R. Balasov, I. Bartos, S. S. Bavera, J. Bellovary, C. P. L. Berry, E. Berti, S. Bianchi, L. Blecha, S. Blondin, T. Bogdanović, S. Boissier, M. Bonetti, S. Bonoli, E. Bortolas, K. Breivik, P. R. Capelo, L. Caramete, F. Catorini, M. Charisi, S. Chaty, X. Chen, M. Chruślińska, A. J. K. Chua, R. Church, M. Colpi, D. D'Orazio, C. Danielski, M. B. Davies, P. Dayal, A. De Rosa, A. Derdzinski, K. Destounis, M. Dotti, I. Duţan, I. Dvorkin, G. Fabj, T. Foglizzo, S. Ford, J.-B. Fouvry, T. Fragkos, C. Fryer, M. Gaspari, D. Gerosa, L. Graziani, P. J. Groot, M. Habouzit, D. Haggard, Z. Haiman, W.-B. Han, A. Istrate, P. H. Johansson, F. M. Khan, T. Kimpson, K. Kokkotas, A. Kong, V. Korol, K. Kremer, T. Kupfer, A. Lamberts, S. Larson, M. Lau, D. Liu, N. Lloyd-Ronning, G. Lodato, A. Lupi, C.-P. Ma, T. Maccarone, I. Mandel, A. Mangiagli, M. Mapelli, S. Mathis, L. Mayer, S. McGee, B. McKernan, M. C. Miller, D. F. Mota,

M. Mumpower, S. S. Nasim, G. Nelemans, S. Noble, F. Pacucci, F. Panessa, V. Paschalidis, H. Pfister, D. Porquet, J. Quenby, F. Röpke, J. Regan, S. Rosswog, A. Ruiter, M. Ruiz, J. Runnoe, R. Schneider, J. Schnittman, A. Secunda, A. Sesana, N. Seto, L. Shao, S. Shapiro, C. Sopuerta, N. Stone, A. Suvorov, N. Tamanini, T. Tamfal, T. Tauris, K. Temmink, J. Tomsick, S. Toonen, A. Torres-Orjuela, M. Toscani, A. Tsokaros, C. Unal, V. Vázquez-Aceves, R. Valiante, M. van Putten, J. van Roestel, C. Vignali, M. Volonteri, K. Wu, Z. Younsi, S. Yu, S. Zane, L. Zwick, F. Antonini, V. Baibhav, E. Barausse, A. Bonilla Rivera, M. Branchesi, G. Branduardi-Raymont, K. Burdge, S. Chakraborty, J. Cuadra, K. Dage, B. Davis, S. E. de Mink, R. Decarli, D. Doneva, S. Escoffier, P. Gandhi, F. Haardt, C. O. Lousto, S. Nissanke, J. Nordhaus, R. O'Shaughnessy, S. Portegies Zwart, A. Pound, F. Schussler, O. Sergijenko, A. Spallicci, D. Vernieri, and A. Vigna-Gómez, "Astrophysics with the Laser Interferometer Space Antenna," *arXiv e-prints*, p. arXiv:2203.06016, Mar. 2022.

[3] V. Korol, E. M. Rossi, P. J. Groot, G. Nelemans, S. Toonen, and A. G. A. Brown, "Prospects for detection of detached double white dwarf binaries with Gaia, LSST and LISA," *MNRAS*, vol. 470, no. 2, pp. 1894–1910, Sep. 2017.

[4] S. F. Portegies Zwart and F. Verbunt, "Population synthesis of high-mass binaries." *A&A*, vol. 309, pp. 179–196, May 1996.

[5] G. Nelemans, L. R. Yungelson, and S. F. Portegies Zwart, "The gravitational wave signal from the Galactic disk population of binaries containing two compact objects," *A&A*, vol. 375, pp. 890–898, Sep. 2001.

[6] S. Toonen, G. Nelemans, and S. Portegies Zwart, "Supernova Type Ia progenitors from merging double white dwarfs. Using a new population synthesis model," *A&A*, vol. 546, p. A70, Oct. 2012.

[7] V. Korol, E. M. Rossi, and E. Barausse, "A multimessenger study of the Milky Way's stellar disc and bulge with LISA, Gaia, and LSST," *MNRAS*, vol. 483, no. 4, pp. 5518–5533, Mar. 2019.

[8] T. B. Littenberg, "Detection pipeline for Galactic binaries in LISA data," *Phys. Rev. D*, vol. 84, no. 6, p. 063009, Sep. 2011.

[9] M. J. C. Wilhelm, V. Korol, E. M. Rossi, and E. D'Onghia, "The Milky Way's bar structural properties from gravitational waves," *MNRAS*, vol. 500, no. 4, pp. 4958–4971, Jan. 2021.

[10] A. Einstein, "Die Grundlage der allgemeinen Relativitätstheorie," *Annalen der Physik*, vol. 354, no. 7, pp. 769–822, Jan. 1916.

[11] s.c. LIGO and c. Virgo, "Observation of gravitational waves from a binary black hole merger," *Phys. Rev. Lett.*, vol. 116, p. 061102, Feb 2016.

[12] V. Korol, N. Hallakoun, S. Toonen, and N. Karnesis, "Observationally driven Galactic double white dwarf population for LISA," *arXiv e-prints*, p. arXiv:2109.10972, Sep. 2021.

[13] C. A. Droppelmann and R. E. Mennickent, "Creating Music Based on Quantitative Data from Variable Stars," *JAAVSO*, vol. 46, no. 2, p. 154, Dec. 2018.

[14] C. Harrison, J. Trayford, L. Harrison, and N. Bonne, "Audio Universe Tour of the Solar System: using sound to make the Universe more accessible," *arXiv e-prints*, p. arXiv:2112.02110, Dec. 2021.

[15] S. Bardelli, C. Ferretti, L. A. Ludovico, G. Presti, and M. Rinaldi, "A sonification of the zCOSMOS Galaxy Dataset," in *HCII 2021: Culture and Computing*. Cham: Springer, 2021, pp. 171–188.

[16] S. Delle Monache, P. Polotti, S. Papetti, and D. Rocchesso, "Sonically augmented found objects," in *Proc. of NIME 08*, Genoa, 2008, pp. 154–157.

[17] D. Willams, "Utility versus creativity in biomedicalmusification," *Journal of Creative Music Systems*, vol. 1, no. 1, pp. 1–13, 2016.

[18] S. Wilson, D. Cottle, and N. Collins, Eds., *The SuperCollider Book*. Cambridge: The MIT Press, 2011.

[19] A. Valle, V. Lombardo, and M. Schirosa, "Simulating the soundscape through an analysis/resynthesis methodology," in *CMMR/ICAD09*, vol. 5954, pp. 330–357.

[20] H. Westerkamp, "Linking soundscape composition and acoustic ecology," *Organ. Sound*, vol. 7, no. 1, 2002.

[21] D. Rocchesso and F. Fontana, Eds., *The Sounding Object*. Firenze: Edizioni di Mondo Estremo, 2003.

[22] A. Valle, *Contemporary Music Notation. Semiotic and aesthetic aspects*. Berlin: Logos, 2018.

[23] T. Sauer, Ed., *Notations21*. New York: Batty, 2009.

[24] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue, "Computer-assisted composition at IRCAM: From PatchWork to OpenMusic," *CMJ*, vol. 23, no. 3, pp. 59–72, 1999.

[25] M. Kuuskankare and M. Laurson, "Expressive Notation Package," *CMJ*, vol. 30, no. 4, pp. 67–79, 2006.

[26] H. Taube, "An introduction to Common Music," *CMJ*, vol. 21, no. 1, pp. 29–34, 1997.

[27] A. Agostini and D. Ghisi, "A Max Library for Musical Notation and Computer-Aided Composition," *CMJ*, vol. 39, no. 2, pp. 11–27, 2015.

[28] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data." in *Proc. of ISMIR 2010*, 2010, pp. 637–642.

[29] A. Valle, "Integrated Algorithmic Composition. Fluid Systems for including notation in music composition cycle," in *Proc. NIME08*, 2008, pp. 253–256.

Sonifying an Office Gadget to Indicate Air Quality

Rod Selfridge
KTH Royal Institute of Technology
rod2@kth.se

Carlo Barone
KTH Royal Institute of Technology
cbarone@kth.se

Sandra Pauletto
KTH Royal Institute of Technology
pauletto@kth.se

ABSTRACT

Poor indoor air quality can produce headaches, fatigue, as well as respiratory and vision issues. Sonification might provide a way to increase people's awareness of their surrounding's air quality and activate preventive actions. In this paper we present the prototype of a sonically augmented interactive office gadget that provides information about air quality when a person interacts with it. We report the results of a user study comparing four sonification models (Abstract, Musical, Concrete and Cultural Models). Results show that the Abstract, Musical and Concrete models are equally effective in displaying air quality levels, while the Cultural model performs less well. Additionally, the Concrete and Musical models are preferred over the Abstract and Cultural models as their sonic qualities are considered more appropriate for what they aim to represent. As a result of this study, we aim to further develop the Musical and Concrete models as well as a Hybrid Model that combines the main characteristics of both.

1. INTRODUCTION

Within workspaces and on desks it is not uncommon to find a number of items that personalise the space. These items could be photographs, plants, trinkets, art to name a few [1]. It has been found that being able to personalise one's own work environment has a positive effect on job satisfaction and well-being. Employee well-being is associated with higher morale, lower absenteeism, and higher productivity [1]. Office gadgets are one type of possible objects which are commonly found in office environments. Examples are: the drinking bird, as described by Frank in [2] or the Newton's Cradle, as described by Gavenda and Edgington in [3].

An investigation on how these objects might enhance the cognitive states of an individual can be found described by Karlesky and Isbister in [4]. This research was further developed in [5] where it was found that fidgeting and fiddling with items around one's own workstation are linked to self-creativity, focus and calm.

The amount of CO2 in offices and indoor spaces has been shown to be associated with symptoms such as fatigue, headaches, respiratory and vision problems [6]. An office gadget that sonically indicates to the user the CO2 levels within their workspace area has obvious benefits for their health and well being.

Copyright: © 2022 Rod Selfridge et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

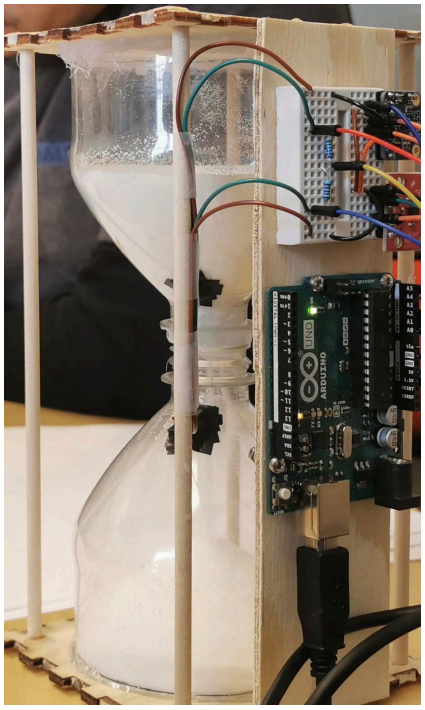


Figure 1. Hourglass augmented to sonify the air quality within an office environment. Photoresistres can be seen attached to the chambers, Arduino and breadboard are attached to the side, with the CO2 monitor and accelerometer on the breadboard.

This paper reports on a project that is part of a wider study into the sonification of air quality. In this project we have augmented an office gadget, a hourglass, with sound in order to communicate the current state of air quality in the room, namely the CO2 level (Fig. 1). Sound models and their ability to communicate the air quality are then examined.

2. BACKGROUND AND RELATED WORK

Extensive research exists into the negative effects of CO2 in working environments [6–9]. The levels of CO2 in a working space appears to be closely related to a phenomenon called *sick building syndrome* (SBS). CO2 levels greater than 800 parts per million (ppm) were found to be linked to an increase in SBS within the office [10]. The main symptoms found were eye irritation and upper

respiratory symptoms. In offices the major contributor to increasing CO2 levels are the occupants, with CO2 being added to the environments when people exhale [11]. CO2 is believed to be an indicator of air pollutants, rather than the direct cause of the observed symptoms [11]. Well ventilated spaces can counteract the effects associated with high CO2 levels [12]. The same study recommended that the CO2 levels be maintained below 1500 ppm, while [13] noted that exposure to 1000 ppm and above can affect cognitive performance.

Art works or objects in a room have previously been used to provide information about the air quality in the environment. An example is the interactive art display of CO2 room levels described by Polli in [14]. The display included a robotic rover which drew a line on the wall relating to the CO2 level in the space. It drew a horizon of "tall and short grasses" illustrating the changing CO2 levels over time. The purpose of this artwork was to promote activism and social interactions. However the paper does not provide strong evidence that these goals were achieved.

Instead of displaying data in a visual manner it can be represented aurally through sonification, defined as *"the use of non-speech audio to convey information. More specifically, sonification is the transformation of data relations into perceived relations in an acoustic signal for the purposes of facilitating communication or interpretation."* [15]. The sonification of data within a shared office environment was reviewed by Vickers in [16]. Here sonification was described as a form of *"serendipitous monitoring"* where attention is focused on a primary task (normal office work) whilst the data is monitored indirectly. A further study by Hermann et. al. [17], highlighted the benefits of this type of sonification compared to traditional alarms and warnings that only become active once a critical threshold has been reached. Sonification might help individuals to indirectly monitor air pollution and take preventative action prior to any such critical levels being reached.

An example of sonification of air pollution was shown by Hall et. al. in [18] where bicycles are augmented with speakers, GPS tracker, an air pollution sensor and a raspberry pi, enabling them to generate audio. The bicycles were presented at Lisboa Soa [1] an environmental sound art festival. Participants were invited to cycle around the city and the data obtained was logged. Researchers were able to build up an air pollution map of the city, while participants were able to perceive through sound the air pollution levels at their current location. Sounds used for the sonifications included the sound of fog horns as well as the sound of panting and gasping. A similar project was presented by Arango et. al. in [19] and [20] where a coat was augmented with air pollution sensors allowing the user to measure the air pollution at their location. Sound pulses (not further described) communicate data by varying their speed and pitch.

A sonification of NO2 [2] and O3 [3] concentration was presented in [21], using filtered noise, sawtooth oscillators and square oscillators. The authors found that untrained listeners could identify air pollution data trends through sonification. The paper identified the need for further work in relation to seasonal trends and geographical positioning.

Three portable devices that could communicate CO2 levels sonically, visually and haptically respectively were presented by Hogan and Hornecker in [22]. The auditory feedback was mapped to the pitch of a synthesised sound, (details of the type of sound are not given). Results showed that subtle changes in CO2 levels were difficult to comprehend from their air pollution data sonification, but all the devices stimulated social engagement about air quality and ventilation.

This project follows on from previous examples of sonification of air quality data but specifically focuses on CO2 levels within an office environment. This research brings together the theme of office space personalisation, which promotes office well-being [1], while allowing users to serendipitous monitor the air quality at their workspace. Such an augmented office gadget may assist with increasing morale and productivity, encouraging a sense of calm but also allowing the user to monitor air pollution and take preventative measures [17], avoiding potential symptoms like fatigue and respiratory issues [6].

Unlike previous research, we focus on determining if users are able to identify levels of air pollution through different sonification mappings of CO2 levels. Further to this, we examine how clearly the sounds represent the air quality data as well as which sounds are preferable. Four different sound models were used, each with their own mappings. Results will then be able to inform future sonifications of air quality data within an office environment.

3. DESIGN

In this project we aimed to augment an office gadget with sound and an air pollution sensor. We investigated a number of office devices that could be easily prototyped and augmented with sound. We looked for objects that one might interact with at times, such as the Newton's cradle, with the aim of augmenting with sound both the normal object behaviour and the data coming from a CO2 sensor attached to it. For our prototype, an hourglass was chosen as a very well understood and common object, that could be easily built with accuracy. A photo of the augmented hourglass is shown in Fig. 1. The object is based on a tutorial given in [23]. The prototype is made using recycled plastic and wood, with salt being used as the grains within the chamber.

3.1 Electronics Components

An Arduino Uno was used to receive data from sensors and communicate with a laptop. Four sensors were used to monitor the state of the object. An Adafruit CCS811 air quality sensor was used to monitor the CO2 level in the space. Two photo-resistors were attached to the upper and lower cavities, indicating when the reservoir of grains had expired. An ADXL377 accelerometer was used to determine when the hourglass was turned.

[1] www.lisboasoa.com
[2] NO2 - Nitrogen Dioxide
[3] O3 - Trioxygen or Ozone

3.2 Sonic Outputs

Sounds were produced on a laptop using Pure Data (vanilla 0.50), with communication from the arduino using the comport object and an additional patch which extracts serial print data [24]. The four sound models - referred to as Abstract, Musical, Concrete and Cultural - were developed along different criteria. The sounds were only audible while the salt was pouring from the top to the bottom chamber. Photo-resistors detected the beginning and end of the flow. This allowed the user to identify when the salt grains had finished falling, giving them the option to interact and turn the hourglass again or not.

Abstract Model: This is a pulsating FM synthesiser model, inspired by an example from Puckette [25]. It was created so that the air quality data linked to the magnitude and frequency of the pulse while the modulation index was controlled by an LFO. When the air quality level is good, the pulsation is approximately 50 - 60 beats per minute, which represents low activity. When the measured air quality decreases the pulse rate and depth increases, designed to give the perception of more activity and effort.

A second FM model was linked to the time taken for the grains to fall from the top chamber to the bottom. The fundamental frequency increases over the time the grains fall with a random variable added making this increase have a meandering quality. The modulation index and frequency are randomised between 15 and 2 times the carrier frequency.

Musical Model: This was created using a bell tone based on an algorithm by Jean-Claude Risset [26]. Historically bells have been used to communicate information (e.g. the sound of the church bells) and alarms. In our model the increase in air pollution equates an increase in the tempo of the bells. The fundamental frequencies of separate drone type tones increases as the grains build up in the bottom chamber.

Concrete Model: This model was developed based on the sound of wind blowing through wires and trees with the assumption that the sound of wind could be easily connected to the idea of air. This model is based on Farnell's wind [27]. The centre frequency of the bandpass filter on a 'wind howling around a wire' sound effect model adjusts with the CO2 level. The wind speed and level of gusts also adjusts with CO2 levels. A separate wire model [27], was linked to the falling of the grains between the chambers. The centre frequency of the bandpass filter used in the model decreases as the grains fall from the top to bottom.

Cultural Model: The CO2 level is connected to a number of frog sound effect models by Andy Farnell [27]. Although triggered randomly, the number of 'croaks' is directly related to CO2 levels. This model was based around a popular saying that describes a hoarse voice as having - *'a frog in the throat'*. The frog croak sound can also be easily associated with someone coughing. Farnell's 'fan' model [27] was adapted so that the speed of the fan decreased as the grains fell from the top to bottom. A demo video [4] shows the hourglass components and sonification.

[4] https://youtu.be/aNTzjXqb5Js

Instance	CO2 level	Comments
Excellent	400	This background level of CO2 is not problematic [11]
Good	415	This level of CO2, a slight increase on the background level of 400PPM, was chosen to signal the start of a deterioration of air quality [11]
Poor	1000	This value was chosen as it was found in [13] that prolonged exposure to CO2 starting at this level affects cognitive performance

Table 1. Relationship between sound instances and CO2 level in PPM

4. METHOD

Research was carried out to investigate the effectiveness of the different sound models, specifically with respect to the clarity of the auditory display of air quality data. We additionally investigated which sound model was preferred by the subjects as the clearest sound model might not be necessarily the preferred. Identifying aspects of clarity and preference can then be used to inform future designs. A small user test was designed to evaluate the prototype and gather feedback about these aspects relating to the four sound models implemented thus far.

4.1 Participants

A total of eight participants (P1 – P8) took part in the test: three female and five males, aged between 23 and 48, with an average age of 31. When asked if participants had personal objects or gadgets on their work desk 6 participants (75%) stated that they did, although only 2 participants out of 6 listed personal items (small skull head, magnetic toys) rather than generic electronics (speakers, tablets, etc). All participants stated that they had some form of musical or sound education. This varied from high school level to PhD in sound and music computing level. Tests were carried out in an office room within KTH Royal Institute of Technology.

4.2 Test Design

For each sound model participants were presented with three different air quality instances: poor, good and excellent (see Table 1). The hourglass was active (grains flowing from top to bottom) when participants were presented with each instance.

A short introduction to the study was provided to all participants prior to the user test commencing. They were told that the hourglass was augmented with audio and that the sound would provide information about air quality in the room. Each of the four sound models were presented to the participants in a random order. For each sound model, three instances of air quality (see Table 1) were presented in a random order to participants. Participants were able to

listen to all the instances and ask for any to be repeated before indicating on a prepared form which air quality level (poor, good, excellent) they believed each sound represented. In total, 24 instances of each sound model were presented during the user test (each participant listened to three instances of each sound model).

Once participants had indicated which air quality the sound instances represented, they were invited to rate, from 1 (completely unclear) to 10 (completely clear), how clearly they thought the sound instances represented the three levels of air quality and their reason for this rating. This shows which of the models communicates the measured air quality best to our participants, and how clear they perceived this communication to be.

Additionally, after participants had heard the three sound instances, they were asked whether they could name one or more sound characteristic that changed as the grains fell from the top to the bottom chamber. Identifying the sound changes over the time of the falling grains gives an indication as to whether the sonic representation of said falling grains was perceived or not. Furthermore, participants were asked whether they could name one or more sound characteristic that they thought related to the air quality to see if they could identify a specific sonic component or if the perceived air quality level was based on a general appreciation of the overall sounds. Finally, after hearing all four sound models, participants were asked to rank the sound models in order of preference, and comment of the reasons of their choices as they may prefer a certain sound even though it is not as clear as another. Any additional comments were noted at the conclusion of the user test.

5. RESULTS

5.1 Abstract Model

For the Abstract model, 2 out of 24 instances were wrongly identified in terms of air quality level (see Fig. 2). P3 identified the good air quality sound instance as poor, and the excellent air quality sound instance as good. The mean rating for clarity for this model was 7.13 out of 10 with a standard deviation of 2.03. A boxplot of these results is shown in Fig. 3.

Amongst comments relating to the clarity rating, P2 stated *"The additional oscillating tones add a sense of urgency. More frequent = More urgent = poorer air quality"*. A total of three participants directly mentioned the frequency of oscillations indicating the change in air quality. P8 commented that *"The changes are clear, with a more subtle sound on the 3rd instance (Excellent)"*. P7 stated they struggled identifying the changes in air quality with this sound model stating that *"It is a bit hard to hear what is positive in these sounds. I can hear the change, but I'm having a bit of a hard time to map it to good or bad air quality"*.

Three out of 8 participants correctly identify sound characteristics changing with the salt falling over time, 4 stated they couldn't and 1 identified the wrong characteristics (see Fig. 4). When asked to describe the characteristics, the successful participants correctly identified the increase

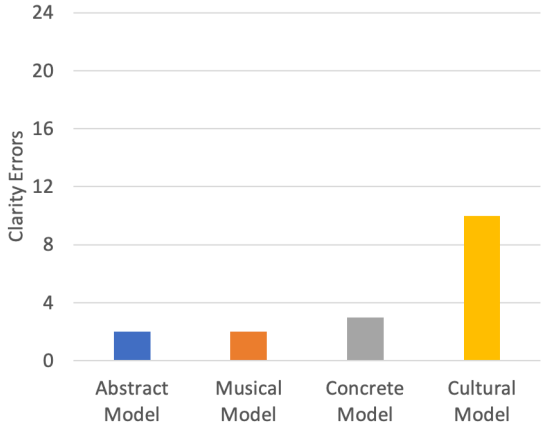


Figure 2. Number of errors when identifying air quality from sound instances for each model

in frequency over time as related to the grains increase in the bottom chamber. P4 stated *"The height of the salt pile affected the higher pitched background noise"*, while P1 stated *"Maybe the glissando in background but I am not very sure about that, sometimes it seems a bit random"*.

Seven out of 8 participants correctly identify sound characteristics changing with the air quality and 1 stated they couldn't, (see Fig. 4). P3 described the frequency of the pulsating sound changing with air quality, P1 stating *"Pulse frequency"*. P4 associated the change in volume with the change in air quality - *"The volume of the pulsating sound"*, while P5 identified both the frequency and volume of the pulsating sound to air quality, *"The sound volume and repetition changes depending on air quality"*.

When ranked against the other models, the Abstract Model was placed in 3rd place by 6 of the participants and in 2nd place by 2 (see Fig. 5). Of those participants ranking this model in 3rd place some of the comments included: *"Very scary, felt like a horror-movie"* (P4), *"I like that the sound is quite subtle, but it is still a bit alarming"* (P8), *" a bit unsettling"* (P7) and *"Gives you a feeling of being watched, space like and mysterious"* (P6). A comment from P1, who ranked this model in 2nd place, was *"Nice background drone, actually I would put this and Sound Model 2 (Musical Model) both as first preference"*.

5.2 Musical Model

For the Musical Model, 2 out of 24 instances were wrongly identified in terms of air quality level (see Fig. 2). The errors from this model were from P3 and P6, both of whom identified the excellent air quality sound instance as good. The mean rating for clarity was 6.38 out of 10 with a standard deviation of 2.33. A boxplot of these results is shown in Fig. 3.

P2 commented that - *"The bell sound adds a level of urgency. As it gets more frequent, it can tell how much poorer the air quality is"*. P8 rated the clarity as 10, stating *"Change in the data was clearly represented in the instances. Increasing alerting"*. In contrast, P6 rated the model for clarity as 3, stating *"They all sounded like quite*

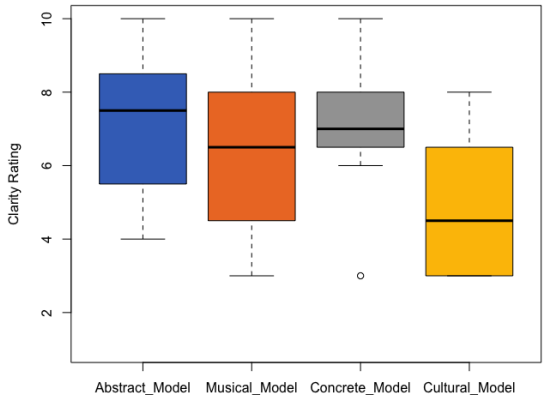


Figure 3. Boxplot showing clarity ratings of air quality sonifications for each model

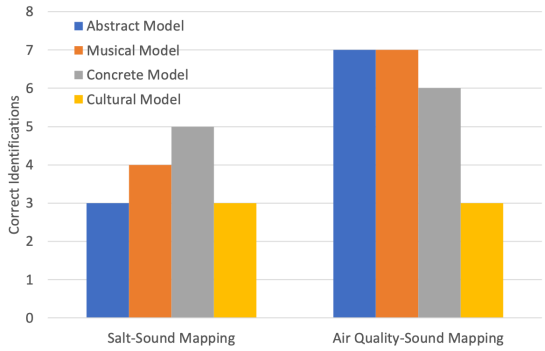


Figure 4. Identification of sound to object mappings

bad air quality".

Four out of 8 participants correctly identify sound characteristics changing with the salt falling over time, 3 stated they couldn't and 1 identified the wrong characteristics (see Fig. 4). When asked to describe the sound characteristics that changed as the grains fell, successful participants identified that the change with the constant drone sounds in the background, although only 1, P2, stated that the *"Frequency of the constant sound"* changed. The others could only identify that something changed with the background sound but did not specify in what manner it changed. P1 stated that *"There is a bell sound each time the 'salt mountain' falls"*, but they were mistaken.

Seven out of 8 participants correctly identify sound characteristics changing with the air quality and 1 stated they couldn't, (see Fig. 4). They all correctly identified that the air quality was related to the bell sound. Comments in relation to the bell sound included *"Intervals between bell sound decreases (faster sounds) with decreasing in air quality. No (bell) sound for excellent air quality"* (P5), and *"Alarm bell warnings increasing in urgency"* (P3). P1 stated *"Rhythm of the bells, fundamental frequency of the drone"*, correctly identifying that the rhythm of the bells changed in accordance to air quality, but mistakenly thinking that the fundamental frequency of the drones changed with the air quality. Similarly, P4 stated *"The frequency of the bells and volume of background drone"*, failing to identify that the volume of the background drones were

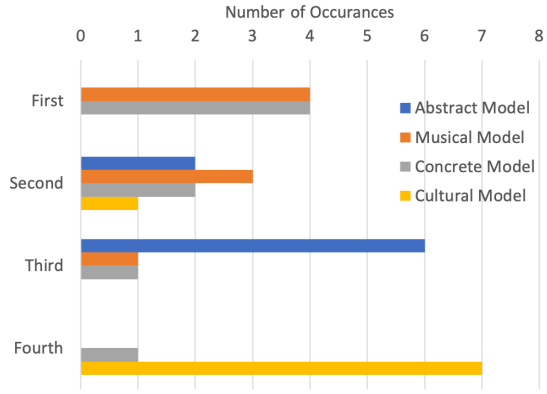


Figure 5. Participants preference of the sound models

changing with the falling sand and independent from the air quality level.

The Musical Model was ranked in 1st place by four participants, followed by 2nd place by another three, and 3rd place by one participant. This is shown in Fig. 5. Of those who rated this model in 1st place, comments included *"Nice bells, sounds meditative"* (P4) and *"I'm partial to the bell sound. It seems fitting to communicate urgency. But maybe have the bell sound appear when it is really poor (air quality)"* (P2). Comments from participants who ranked this model in 2nd place included *"The bells provide a clear signal for changing quality, and I want them to stop"* (P8), and *"Calming but still alerting. Would be annoyed with the bells after a while"* (P7).

5.3 Concrete Model

For the Concrete Model, 3 out of 24 instances were wrongly identified in terms of air quality level (see Fig. 2). The errors from this model were from P1, P3 and P6, all of whom identified the excellent air quality sound instance as good. The mean rating for clarity of this model was 7 out of 10 with a standard deviation of 2.07. A boxplot of these results is shown in Fig. 3.

Four participants stated that the change in volume allowed them to identify changes between the instances. P7 stated *"Easy to spot a difference between the instances, and for me the increase volume represent a poorer air quality"*, and likewise, P5 stated *"The volume increases as the air quality reduces. Also feels like the speed is increasing as the quality gets worse"*. Some possibility of confusion was noted in the participants comments: *"If the changes are too slow, there's a chance that some people wouldn't be able to notice"* (P2) and *" Sound instance 3 (Poor) can either be good air quality, because it is windy, fresh air, or bad because in could be a fan"* (P6).

Five out of 8 participants correctly identify sound characteristics changing with the salt falling over time, 2 stated they couldn't and 1 identified the wrong characteristics (see Fig. 4). The successful participants correctly identified the decrease in pitch of one of the howling sound as the grains fall - *"Pitch goes down over time / with more salt"* (P8). P7 identified the characteristic that changed over time, but not how it changed, *"The high pitch sound,*

not the white noise. It matched the salt in the hourglass well. If the hourglass could make a sound in would be that kind of whistle". P4 and P5 were not able to identify the characteristic that changed, stating *"It sounds a little bit like the flow of salt affects the noise"* and *"Humming sound which might correspond to the flowing salt"* respectively.

Six out of 8 participants correctly identify sound characteristics changing with the air quality, 1 stated they couldn't and 1 identified the wrong characteristics (see Fig. 4). The most common characteristic identified with air quality was volume, or level, increasing as the air quality decreased. Examples of comments are: *"The high pitched sound increased in volume when quality dropped"* (P4), *"The white noise increasing. White noise = wind = Air = increasing when bad air quality"* (P7) and *"Overall level, and the level (and perhaps frequency) of the middle whistling sound"* (P2). P1 believed the bandwidth of a filter was being altered between sound instances which was not the case.

The Concrete Model was placed in 1st place by four participants, 2nd place by two participants, with the remaining participants ranking this sound model as 3rd and 4th (see Fig. 5). Comments from participants that ranked this model in 1st place include - *"The quiet sound was nice, felt positive"* (P6), *"Sounds like wind and is quite subtle"* (P8), and *"This one I could see myself use in an office. The white noise + whistle is subtle but I would notice a change"* (P7). P1 ranked this in 3rd place stating *"The sound reminded of whistles and winds that become annoying after a while"*.

5.4 Cultural Model

For the Cultural Model, 10 out of 24 instances were wrongly identified in terms of air quality level (see Fig. 2). Only P5, P7 and P8 made no errors when presented with the different sound instances. P2, P3 and P6 identified all the sound instances as poor, while P1 identified the good air quality sound instance as poor and the excellent air quality sound instance as good. P4 identified the excellent air quality instance as poor and the poor air quality instance as excellent. The mean rating for clarity of this model was 4.88 out of 10, with a standard deviation of 1.96. A boxplot of these results is shown in Fig. 3.

Participants indicated that it was difficult to identify the difference between good and poor. For example, P5 stated *"Instance 1 (Good) and instance 2 (Poor) might be confusing as compared to instance 3 (Excellent)*, and P4 stated *"Hard to tell the difference between 1 (Poor) and 2 (Good)"*. A similar comment was given by P7 *"Didn't notice much changes between instance 1 (Poor) and 2 (Good), and the overall soundscape felt a bit discomforting, making it hard to map 'good' or 'excellent' "*. This feeling of discomfort was highlighted by a number of other participants - *"The background noise makes all sounds feel bad"* (P6), *"The constant sound felt really unpleasant"* (P2), and *"The consistent low drone is a bit creepy / eerie, this suggests to me that in general something is wrong"* (P1).

Three out of 8 participants correctly identify sound characteristics changing with the salt falling over time, 4 stated they couldn't and 1 identified the wrong characteristics (see Fig. 4). Out of these, only Participant 8 identified that there was a decrease in pitch of the constant sound as the grains fell. P5 and P7 both identified that the humming, background noise corresponded to the flow of grains, but did not specify in what way the sound characteristics changed.

Three out of 8 participants correctly identify sound characteristics changing with the air quality and 5 stated they couldn't (see Fig. 4). P4 and P5 identified the frequency of frog sounds related to the air quality. P4 correctly identified the frog sounds as relating to the air quality, but perceived an increase in these sounds as an increase in air quality.

The Cultural Model was place in 4th place by all participants but one (see Fig. 4). P5 ranked this model in 2nd place but stated *"Sound might be too distracting"*. Other comments were *"less interesting* (P1), *"too annoying"* (P2) and *"less indicative"* (P3). P6 and P7 expressed negative comments about the model stating *"Felt very dark"* (P6) and *"Quite unsettling. Reminded me of a dark forest"* (P7).

6. DISCUSSION

In terms of clarity, Abstract, Musical and Concrete models seem to perform equally well (see Fig. 2). Participants intuitively understand the meaning of the sonification mapping and some are even able to identify the sound characteristics linked to air quality and the salt pouring. The Cultural model performs between 3 and 5 times less well indicating that participants found identifying air quality levels from this model's sounds a greater challenge.

A number of reasons for the Cultural Model's poor clarity have been identified from participants comments. Firstly, the fan sound effect that slows down as the sand grains pour was found to be disturbing by participants. It was mistaken as *"machinery"* (P8), *"buzzsaw"* (P4) or *"helicopter"* (P2), none of which are associated with clean air. The triggering of the frog sound effect did not communicate the air quality clearly enough. The 'croaks' were triggered by a random number generator and the frequency used to trigger the generator was linked to the air quality. This had the effect that the generation of the croak sounds could be sporadic no matter what the air quality.

Finally, the combination of the frog sound and the fan sound was not perceived as pleasant, being described as *"very dark"* (P6) and *"unsettling"* (P7). Additionally, the frog sound could be interpreted in completely the opposite way it was intended as demonstrated by P4 who thought that more frogs equated to a cleaner, greener environment. Additionally, P4 interpreted that when the frog 'croaks' subsided, they were *"dying from bad air"*. When explaining the models to all participants, at the end of the study, it also became apparent that none made the connection between the frog sound model and the saying *'Frog in your throat'* despite the fact that a number of people were familiar with the saying. The frog sound effect was not associated with coughing either. We conclude that these factors combined produced the result that the Cultural Model

performed much worse than the other three.

The mappings of the Abstract, Musical and Concrete Models characteristics to air quality and the salt pouring seem to have been overall more intuitive and clear, despite the fact that many participants were not able to completely identify which sound characteristics were linked to the pouring salt grains or the air quality. This is not a significant issue as the sonification should work intuitively, without requiring from the user learning or have detailed knowledge of what is happening. What is of importance here is the fact that participants were able to discern the different instances of excellent, good and poor air qualities overall.

In terms of preference, the Musical and the Concrete Models are clear favorites (see Fig. 5) with the Abstract Model in third place, and the Cultural Model last on fourth place. A Kruskal-Wallis Rank Sum Test shows that the Musical Model is significantly different ($p = 0.006$) than an average ranking (2.5) as is the Cultural Model ($p = 0.005$). Although the Abstract, Musical and Concrete Models all have high ratings for clarity, participants had more negative cultural associations with the FM sound (see comments such as (*"felt like a horror-movie"* (P4), *"a bit unsettling"* (P7), *"a feeling of being watched"* (P6))) which might be the reason why the Abstract Model ranked below the Musical and Concrete Models.

Contrary to the negative associations with the Abstract Model, the Musical Model and Concrete Model both had more positive associations. The bells were described as *"meditative"* (P4) and *"Calming but still alerting"* (P7) however some comments suggested that the bell sound might become tiresome over time - *"a clear signal for changing quality, and I want them to stop"* (P8).

Similar comments were made in relation to the Concrete Model, received remarks included - *"felt positive"* (P6), *"is quite subtle"* (P8) and *"This one I could see myself use in an office"* (P7). The Concrete Model also attracted some criticism with both P1 and P4 describing the model as *"annoying"*. P2 also highlighted that the higher whistling tone *"will be annoying for people with tinnitus"*, which is an aspect that should considered in further prototypes.

7. CONCLUSION

We have described the design of a sonically augmented interactive office gadget that aims to provide information about the room air quality to its user. We evaluated four different sound models (Abstract, Musical, Concrete and Cultural Models) via a user study. Results showed that the Musical and Concrete models are both the clearest as displays of information and the preferred models sonically.

A number of additional aspects should be considered in future developments. On the physical object, it is believed that force sensitive resistors below the grains would provide a better measure of the flow of grains. A number of photo resistors were intended to be used in this initial prototype, but we found that their sensitivity to light in the environment made them difficult to program with any accuracy (measurements varied depending on lighting in different office spaces and even time of day).

Different physical objects might work better with the different sound models. For example, the Abstract Model might gain a higher preference rating for some people if connected to a more futuristic or 'dark' object.

As a result of this study, three sound models will be brought forward to the next development stage of the augmented hourglass: the Musical Model, Concrete Model and a Hybrid Model combining Musical and Concrete characteristics. This Hybrid Model is inspired by a comment from P2 *"Personal preference: wind sound from model 3 (Concrete Model) plus bell sound from model 2 (Musical Model). The combination makes the sound similar to a common city soundscape"*.

Furthermore, it would be of great benefit to carry out a longitudinal user-study, observing participants use of the object over a more substantial period of time. This would give a greater understanding of how the sounds integrate into the office environment, and if annoyance can occur with an increased exposure.

Finally, a number of different environments could be tested (e.g. home, vehicles, elevators). For instance it is known that poor air quality in vehicles is linked to negative cogitative effects [28], therefore a similar augmented object for vehicles could have the potential to make a significant difference in these environments.

8. REFERENCES

[1] M. M. Wells, "Office clutter or meaningful personal displays: The role of office personalization in employee and organizational well-being," *Journal of environmental psychology*, vol. 20, no. 3, pp. 239–255, 2000.

[2] D. L. Frank, "The drinking bird and the scientific method," *Journal of Chemical Education*, vol. 50, no. 3, p. 211, 1973.

[3] J. Gavenda and J. R. Edgington, "Newton's cradle and scientific explanation," *The Physics Teacher*, vol. 35, no. 7, pp. 411–417, 1997.

[4] M. Karlesky and K. Isbister, "Designing for the physical margins of digital workspaces: fidget widgets in support of productivity and creativity," in *Proceedings of the 8th international conference on tangible, embedded and embodied interaction*, 2014, pp. 13–20.

[5] M. Karlesky and K. Isbister, "Understanding fidget widgets: Exploring the design space of embodied self-regulation," in *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*, 2016, pp. 1–10.

[6] M. G. Apte, "Associations between indoor CO2 concentrations and sick building syndrome symptoms in us office buildings: an analysis of the 1994-1996 base study data," *Indoor air*, vol. 10, no. 4, 2000.

[7] K. Permentier, S. Vercammen, S. Soetaert, and C. Schellemans, "Carbon dioxide poisoning: a literature review of an often forgotten cause of intoxication

in the emergency department," *International journal of emergency medicine*, vol. 10, no. 1, pp. 1–4, 2017.

[8] Z. Aghalari, A. Amouei, A. Zarei, M. Afsharnia, Z. Graili, and M. Qasemi, "Relationship between CO2 concentration and environmental parameters with sick building syndrome in school and house settings in babol, iran," *Journal of Mazandaran University of Medical Sciences*, vol. 29, no. 171, pp. 31–44, 2019.

[9] D. P. Wyon, "The effects of indoor air quality on performance and productivity." *Indoor air*, vol. 14, pp. 92–101, 2004.

[10] D.-H. Tsai, J.-S. Lin, and C.-C. Chan, "Office workers' sick building syndrome and indoor carbon dioxide concentrations," *Journal of occupational and environmental hygiene*, vol. 9, no. 5, pp. 345–351, 2012.

[11] X. Zhang, P. Wargocki, Z. Lian, and C. Thyregod, "Effects of exposure to carbon dioxide and bioeffluents on perceived air quality, self-assessed acute health symptoms, and cognitive performance," *Indoor air*, vol. 27, no. 1, pp. 47–64, 2017.

[12] T. Vehviläinen, H. Lindholm, H. Rintamäki, R. Pääkkönen, A. Hirvonen, O. Niemi, and J. Vinha, "High indoor CO2 concentrations in an office environment increases the transcutaneous CO2 level and sleepiness during cognitive work," *Journal of occupational and environmental hygiene*, vol. 13, no. 1, pp. 19–29, 2016.

[13] K. Azuma, N. Kagi, U. Yanagi, and H. Osawa, "Effects of low-level inhalation exposure to carbon dioxide in indoor environments: A short review on human health and psychomotor performance," *Environment international*, vol. 121, pp. 51–56, 2018.

[14] A. Polli, "Breathtaking," in *Proceedings of the Media ecology association*, vol. 11, 2010, p. 7.

[15] G. Kramer, B. Walker, T. Bonebright, P. Cook, J. H. Flowers, N. Miner, and J. Neuhoff, "Sonification report: Status of the field and research agenda," 2010.

[16] P. Vickers, "Sonification for process monitoring," in *The sonification handbook*. Logos Verlag, 2011, pp. 455–492.

[17] T. Hermann, T. Hildebrandt, P. Langeslag, and S. Rinderle-Ma, "Optimizing aesthetics and precision in sonification for peripheral process-monitoring." Georgia Institute of Technology, 2015.

[18] K. M. Lisa Hall and F. Visi. (2021) Environmental bike. [Online]. Available: www.sonicbikes.net/environmental-bike-2020/

[19] M. P. G. Arango and J. J. Arango, "Design process for wearable technologies and urban ecology, airq jacket," in *Proceedings of the 23rd International Symposium on Electronic Art (ISEA) 2017*, 2017, pp. 443–452.

[20] J. J. Arango, "Airq sonification as a context for mutual contribution between science and music." *Revista Música Hodie*, vol. 18, no. 1, 2018.

[21] J. L. Laughner and E. K. Canfield-Dafilou, "Illustrating trends in nitrogen oxides across the united states using sonification," in *The 23nd International Conference on Auditory Display, Pennsylvania, USA*, June 2017.

[22] T. Hogan and E. Hornecker, "Feel it! see it! hear it! probing tangible interaction and data representational modality," in *Proceedings of DRS International Conference, Brighton, UK. 27–30 June,*, 2016.

[23] C. Channel. (2021) How to make a hourglass using plastic bottle. [Online]. Available: www.youtube.com/watch?v=uG0ubTx2jMw

[24] A. Drymonitis. (2021) Arduino pd. [Online]. Available: www.github.com/alexdrymonitis/Arduino_Pd

[25] M. Puckette, *The Theory and Techniques of Electronic Music*. World Scientific Publishing Company, 2007.

[26] C. Dodge and T. A. Jerse, *Computer music: synthesis, composition, and performance*. Macmillan Library Reference, 1985.

[27] A. Farnell, *Designing sound*. Mit Press, 2010.

[28] N. Hudda and S. Fruin, "Carbon dioxide accumulation inside vehicles: The effect of ventilation and driving conditions," *Science of The Total Environment*, vol. 610, pp. 1448–1456, 2018.

A Perceptual Study of Sound Ecology in Peripheral Sonification

Maxime Poret
Univ. Bordeaux,
Bordeaux INP, CNRS, LaBRI,
UMR5800, F-33400 Talence, France
maxime.poret@labri.fr

Catherine Semal
Univ. Bordeaux,
Bordeaux INP, CNRS, INCIA,
UMR5287, F-33076 Bordeaux, France
catherine.semal@ensc.fr

Myriam Desainte-Catherine
Univ. Bordeaux,
Bordeaux INP, CNRS, LaBRI,
UMR5800, F-33400 Talence, France
myriam@labri.fr

ABSTRACT

Based on a case study on 3D printing, we have been experimenting on the sonification of multidimensional data for peripheral process monitoring. In a previous paper, we tested the effectiveness of a soundscape which combined intentionally incongruous natural and musical sounds. This was based on the hypothesis that auditory stimuli could better stand out from one another if they were less ecologically coherent, thus allowing for better reaction rates to various notifications. In this paper, we follow up on that hypothesis by testing two new acoustic ecologies, each exclusively consisting of either musical or natural sounds. We then run those ecologies through the same dual-task evaluation process as the previous one in order to compare them. The results seem to favor our hypothesis, as the new ecologies were not detected as accurately as the original. Though, the set of natural sounds seemed to be considered less intrusive by testers, and to allow for a better performance at an external primary task. We hope to see this work become part of a much larger corpus of studies, which may eventually provide a more definite answer on the effect of ecological coherence in peripheral soundscape design.

1. INTRODUCTION

The display of data through non-verbal sounds, or sonification [1], has been shown to adequately provide various types of notifications for process monitoring tasks by Gaver et al. [2], then Rauterberg et al. [3], even for large numbers of monitoring criteria. Although historically it has been used for decades to convey alerts in industrial processes (often in the overly-stressful form of bells, whistles, honks and beeps), the advent of more advanced sound and data processing technologies in recent years has brought on an effort to move on from the "better safe than sorry" paradigm [4] towards calmer, less invasive soundscapes [5]. In this paper, we focus on the "peripheral" kind of sonification, which should give listeners access to various auditory notifications on data states while also allowing them to take care of more attention-demanding primary tasks [6].

Copyright: © 2022 Maxime Poret et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Traditionally, the design guidelines for such sonifications recommend using a small collection of concurrent, continuously playing sound streams [7], which should be selected with ecological coherence in mind. This means both making sure that the streams are not masked by sounds otherwise present in the use context, and ensuring that the spectral bandwidths of the sounds are allocated in a way that prevents them from masking one another [3, 6, 8]. Our goal here is not to question those precautions against masking, as they seem quite self-evident. However, sonification designers often appear to follow an extra rule that this coherence should also involve a particular attention to logical likelihood in the sound choices, so that soundscapes usually revolve around themes (or "ecologies") such as "forest", "beach", or various forms of exclusively musical displays.

Recent research into auditory processing suggests that the discrimination of sounds happens on a neurophysiological level through a hierarchy of parallel classifications [9–11], and that natural sounds, even highly shortened and simplified, can still be recognized amidst incongruous sets of musical sounds [12]. We wonder, then, if it might be advantageous to select sounds belonging to very different cognitive categories (e.g. musical sounds vs. animal cries) so that cortex-level classification processes may better separate them.

In a previous work on process monitoring sonification [13], we formulated the hypothesis that an incongruous set of stimuli might allow for a faster and more accurate discrimination of notifications than a logically coherent one. Our goal with this work is to conduct an experiment testing this hypothesis.

It is quite common for researchers studying process monitoring sonification to propose several sets of stimuli based on different design strategies (often a dichotomy of auditory icons versus musical mappings) [14–18]. However, works including a "mixed" ecology of both musical and natural sounds seem to be more difficult to come by [8, 19] and none, to our knowledge, has produced a full comparative evaluation of those design choices beyond qualitative user surveys.

In this work, we designed and evaluated two ecologically coherent sets of stimuli in order to compare them to an incoherent one we previously tested. To this end, we reused some of the existing sound streams and completed them with new ones, which were selected according to the respective ecological guidelines of the new sets (Section 3).

We then evaluated them using the same dual-task experiment as before (Section 4) so that we could compare, for each ecology, the quality of anomaly annotation, the effects on primary task performance, and the listeners' aesthetic appreciations (Section 5).

2. CONTEXT

2.1 3D Printing

This project stems from a case study on the use of sonification to help monitor the wire arc additive manufacturing process [20]. During this process, metallic parts are built up layer by layer by the addition of material which gets heated up, melted and deposited by bolts of high-voltage current running through a wire. As such a process tends to be noisy, flashy, and overall unpleasant to be around (let alone monitor directly), there has been an effort in the past few years to augment its observation through the use of alternative modalities, such as haptic vibrations [21], visual augmented reality [22] or, in our case, a peripheral auditory display [13, 23].

The peripheral aspect of this display is expected to provide operators with a more remote form of monitoring, which would allow them to move away from the printer and take care of other tasks such as checking their e-mail, managing their schedules, handling other machines, etc., while still being able to come back to the process in reaction to notifications.

The manufacturing criteria that our display should help monitor are described below. See also the first column of Table 1 for the full detail of thresholds and tolerances.

- The Weld Pool Dimensions (WPD) are the width and height of the material deposit right below the printing head.
- The Part Height (PH) is the cumulative height of all the constructed layers below the printing head.
- The Weld Pool Temperature (WPT) is the local temperature of the material deposit.
- The Part Temperature (PT) is the average temperature of the part being constructed.

2.2 Simulated Process

As of writing this article, the sensors which would be able to provide those monitoring data are still being researched [24]. Instead, our sonification is produced by running SuperCollider [1] scripts on synthetic log files which emulate the way criteria should behave during the printing process.

In order to properly evaluate the sounds-to-criteria connections, one would need to account for users' learning curve over a proper training phase, so the experiment would need to take place on a relatively long term (several trials a week over a month or two). This would also ideally require a population of printer operators, who would already be well-acquainted with the printing process and familiar with its conditions and criteria.

[1] https://supercollider.github.io/

In the current state of this project, such in-situ experimentation is not yet feasible. So, we decided not to concern ourselves with studying the mental connections between sound metaphors and manufacturing criteria yet, and to instead focus our experimental process on the detection of the sounds themselves, as well as their usability in a (simulated) peripheral work context. Despite this limitation, we hope to lay the groundwork for future larger-scale experiments studying the use of similar displays in real monitoring situations.

3. SOUND MAPPINGS

3.1 General Principles

Even though the association of sounds to criteria is not studied here, we want to be prepared for that next step by carefully selecting metaphors that relate to the different types of physical data.

For our natural stimuli, we avoided selecting concurrent sounds that would be too difficult to differentiate without any visual context. Indeed, for example, the sounds of heavy rain, boiling water, or a flowing river, may easily be confused for one another, or mask each other when occurring at the same time. Likewise, when selecting our musical stimuli, we made sure that the streams had non-overlapping pitch ranges, as well as different timbres and granularities.

The choices of timbre, instrument, or sound source vary from ecology to ecology, but we aim to maintain as many psychoacoustic concerns as possible across all ecologies. The thematic dimension is the main variable here. As such, the soundscape always has an "idle" state made up of two continuous stimuli indicating normal WPD and PH values. Both dimensions for WPD are mapped to three parameters (pitch for height, rhythm for width, and loudness for both), defining a slowly-repeating burst of sound, which metaphorically represents the progressive accumulation of material. PH is conveyed by a continuous sound, which can evolve into three states (normal, too low, too high) and represents the base upon which the material is being deposited. WPT is conveyed by silence in normal situations, but can emerge in one of two forms depending on the direction of the anomaly. Indeed, there is no need to constantly remind listeners that the temperature is right, especially if we want to avoid overloading the display in the absence of anomalies. PT is conveyed by a single sound event denoting the passing of the threshold. It is a last-resort alarm which we choose to make conceptually related to the sound for WPT, as the global temperature alarm is expected to be brought on by a prolonged local temperature anomaly.

The following subsections give a description of the mapping choices for each ecology. See Table 1 for an overview of those mappings for each stimulus and how they combine to form each of the three ecologies.

3.2 Mixed Ecology

As previously described in [13], the Mixed ecology was designed so that geometrical (WPD, PH) and thermal (WPT, PT) criteria would be conveyed by two distinct categories

WPD Width: $4mm \pm 10\%$ Height: $3mm \pm 10\%$	Height difference	Tonic pitch: C5 (normal) to F6 (worst)	Arpeggio (Mixed & Synth)
	Both differences	Loudness: 0.02x to 0.2x	
	Width difference	Inter-offset interval: 0.5 to 1.5 s	
	Height difference	Playback speed: 0.5x to 2.0x	Droplets (Nature)
	Both differences	Loudness: 0.1x to 0.8x	
	Width difference	Intervals: 0.5 to 1.5 s	
PH Current layer $\pm 1.5mm$	Relative difference	Pitch: A3 $\pm$3 tones	Drone (Mixed & Synth)
	Absolute difference	Loudness: 0.1x to 0.4x	
	Relative difference	Mixing: Ducks, Misc. or Crows	Birds (Nature)
	Absolute difference	Loudness: 0.0x to 0.3x	
WPT $2000°C \pm 10\%$	Difference polarity	Pitch: 220 Hz or 880 Hz	Jingle (Synth)
	Absolute difference	Loudness: 0.0x, 0.1x or 0.2x	
	Difference polarity	Selection: Crackling or Boiling	Water (Mixed & Nature)
	Absolute difference	Loudness: 0.0x, 0.2x or 0.5x	
PT $600°C$	Threshold	Trigger: Sound sample	Bell (Synth)
		Trigger: Random components	Sizzle (Mixed & Nature)

Table 1. Overview of the mappings from each monitoring criterion (left) to each stimulus (right). In parentheses after each stimulus name: the names of the ecologies in which it is involved.

of sounds, respectively based on auditory icons [25] and abstract earcons [26] guidelines.

The thermal criteria were relatively easy to convey through metaphorical auditory icons, as there are many real-world phenomena whose sounds directly relate to the idea of temperature (boiling water, crackling ice, burning fire, sizzling water). The geometrical criteria, however, required further levels of abstraction in order to be conveyed by sounds. Indeed, it is rather difficult to directly correlate natural phenomena to the concepts of "width" or "height". For this reason, we instead chose to map those dimensions to perceptual parameters in musical streams.

The Mixed ecology thus consists of the 4 stimuli Arpeggio (WPD), Drone (PH), Water (WPT) and Sizzle (PT), as detailed below.

Arpeggio is a looping motive of 3 pitches (tonic, third and fifth) in the major scale, played by the default SuperCollider synth, a simplistic piano-like sound. In idle state, this sound stream consists in a repetition of a low-volume C5 Major arpeggio, with each note separated by 1.5 seconds. When the tonic pitch, duration, or loudness is changed by a fluctuation in WPD (see mappings in Table 1), the newly-defined motive interrupts the one previously started.

Drone is a continuous bandpass-filtered sawtooth wave playing a fluctuating pitch. In idle state, it plays a constant low-volume A3 note. From there, its pitch can get lower or higher following the direction of the PH discrepancy.

Water can be seen as a small body of water which is idle and silent most of the time, but can start boiling or freezing as a result of unnaturally quick changes in temperature.

As a continuation of this Water metaphor, the Sizzle alarm consists in a more violent evaporation sound, reminiscent of water being poured onto a red-hot surface.

3.3 Synth Ecology

For the Synth ecology, we only chose musical sounds, using a variety of synthetic timbres with abstract mappings to perceptual parameters. We reused Arpeggio and Drone as defined for Mixed and designed two new stimuli for thermal dimensions WPT and PT.

For WPT, the natural metaphor selection was replaced by a binary mapping to the pitch of a rapid percussive sound grain (a repeating 0.06s sine wave burst with quick attack and decay), which we call "Jingle". This stimulus is silent in idle state. It emerges in loudness as an anomaly arises, and its pitch can take one of two values (220 Hz for underheating and 880 Hz for overheating).

The PT threshold alarm was replaced by "Bell", an inharmonic sound with quick attack and slow decay reminiscent of a large bell being struck. Like previously with Sizzle and Water, this metaphor was chosen as a continuation of the Jingle timbre. We boosted this alarm's novelty aspect for listeners by making its timbre partially randomly-generated: around a constant 440 Hz main frequency, 3 inharmonic components are randomly selected between 220 and 880 Hz.

3.4 Nature Ecology

The Nature ecology is based on the principles of auditory icons design. We reused the Water and Sizzle auditory icons previously defined for Mixed and added two new stimuli for geometrical dimensions WPD and PH, using metaphors akin to a "pond in the forest" ecological theme.

The Nature ecology thus consists of the stimuli Droplets (WPD), Birds (PH), Water (WPT) and Sizzle (PT).

As noted in the description for the Mixed ecology, it is difficult to select natural sound metaphors which can be related to the geometrical dimensions of an item. But we can benefit from a property of natural sound processing,

which Gaver calls "everyday listening" [27], by which listeners detect, from the perceptual properties of a sound, the characteristics of the object or action producing it. For instance, a larger object falling into a pond produces a lower-pitched and longer "splash" sound than a smaller one. It is by a similar logic that we defined the Droplet stimulus, which conveys WPD using a short sound sample of a drop of water falling onto a surface. The height was mapped to its playback rate so that it would sound higher-pitched if the weld pool became higher (as if falling harder), and the width was mapped to its occurrence rate so that it would be heard more often as the weld pool became wider (as if overflowing).

The Birds stimulus conveying PH is based on the idea of constructing a continuous layer of miscellaneous birdsong, such that in idle state no information stands out from an overall pleasant and familiar noise. In case of an anomaly, a single species sound emerges, thus breaking the noise and calling for the listener's attention. That species is either a duck if PH is too high or a crow if it is too low.

This abstract metaphorical choice of ducks or crows to translate height is an attempt to map that dimension to the respective affective connotations of those birds in Western folklore: the duck is a familiar domesticated bird (positive opinion, so a "high" polarity), while the crow is a scavenger and a bad omen (negative opinion, "low" polarity).

4. EXPERIMENT

In order to preserve the experimental conditions for comparability sake, and since it provides a basic simulation of a peripheral monitoring work context, we reused the same dual-task game that was originally built to evaluate the Mixed ecology. See Figure 1 for a screenshot of the experimental interface. The primary task consists in a very simple sequence-copying task using the mouse, while the secondary task consists in listening for notifications in the soundscape and annotating them as they occur by checking boxes on the side.

Due to sanitary constraints at the time of the experiment, the evaluation game had to be sent out in the form of an interactive web page using JavaScript and PHP elements to log user actions during testing. Each participant took the test in their own environments and on their own setups, and as we acknowledge this potential lack of uniformity, we hope it can be compensated for by a large enough population sample of testers. Both sides of the experiment, featuring either one of the new ecologies, are implemented on the same page such that, upon user connection, one or the other set of stimuli is selected depending on which one has been the most represented in completed tests so far. If both ecologies have been tested the same number of times, one is picked randomly.

Participants start the experiment by undergoing a training phase during which they are introduced to the primary task, then each sound stimulus in isolation, then a few examples of the full soundscape in idle and anomalous states. Finally they have to successfully complete two levels ensuring they understood the process before moving on to the actual experiment, which consists of ten 30-seconds levels

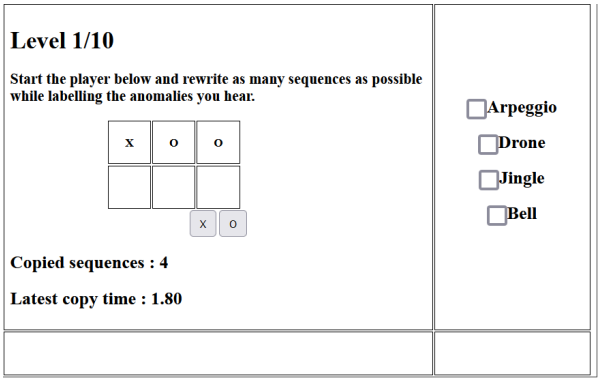


Figure 1. A screen capture of the experiment interface during a level, here for the Synth ecology. In the middle, the player rewrites the sequence displayed on the upper row by clicking the 'X' and 'O' buttons in the same order. Upon each sequence completion, a new one appears. Boxes on the right allow the player to label anomalies as they occur. The labels on these boxes depend on the names of the stimuli in use for the ecology being tested.

featuring various anomaly combinations selected in a random order.

The application logs the times at which anomaly boxes are checked over the course of a level, as well as the size and time of completion of each sequence copied.

Upon completing the experiment, participants are redirected towards a brief survey, first asking them for the usual demographic information (age and gender) before presenting them with a series of statements regarding the exercise (e.g. "the task was hard", or "the sound was stressful", see Table 2) to which they can reply with "Disagree", "Somewhat agree", and "Agree". Finally, a free-form comment box allows them to further develop their feedback.

5. RESULTS

Synth was tested by 15 participants. Four of them had taken part in an earlier iteration of the experiment. 8M, 5F, 2 not specified. Ages 20 to 69 ($\mu = 35.9, \sigma = 15.9$).

Nature was tested by 16 participants. Two of them had taken part in a previous iteration of the experiment. 7M, 9F. Ages 19 to 62, ($\mu = 24.75, \sigma = 10.5$).

In the previous experiment, Mixed had been tested by 43 participants. 20M, 23F. Ages 18 to 67 ($\mu = 32.3, \sigma = 12.4$).

We used the data collected during the experiment to compute an array of quality metrics for the displays, detailed in the following subsections: the annotation accuracy for each stimulus, the reaction time needed to label anomalies, the distribution of primary task performances, and the qualitative user feedback.

5.1 Annotation Accuracy

Signal Detection Theory [28] allows us to compute a metric called sensitivity index, or d', using the hit (H) and false alarm (FA) rates of each participant in the annotation task,

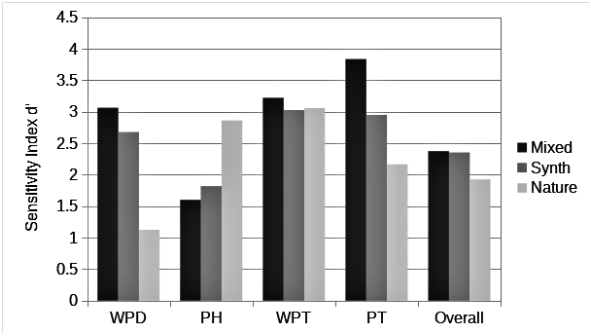


Figure 2. Mean sensitivity index for each stimulus type (horizontal sections) and for each ecology (colors).

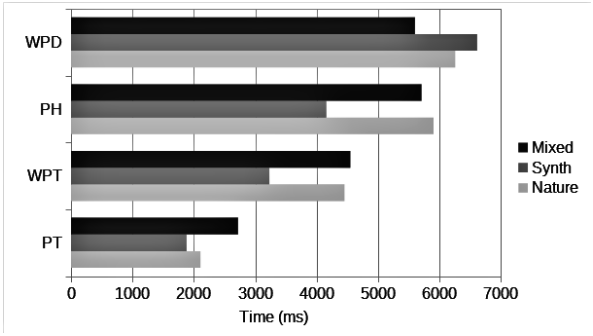


Figure 3. Average annotation times in milliseconds for each stimulus type (vertical sections) and for each ecology (colors).

such that $d' = Z(H) - Z(FA)$, with Z the inverse function of the cumulative normal law distribution. We applied the "mean sensitivity" strategy by averaging all the participants' d' indices for each stimulus type. Since some users' rates were too "exact" to allow for Z computation (total rates of 0 or 1), we approximated them to 10^{-2} of their actual values (respectively resulting in 0.01 and 0.99). This is equivalent to considering that, if given a hundred trials for each stimulus, the most efficient testers may still have a 1% chance of making a mistake.

This sensitivity index is such that a higher value denotes a stronger intelligibility for the corresponding stimulus. Since it has no dimension or scale, it cannot give an absolute indication of the quality of a signal (beyond the fact that a signal with $d' > 1$ has above-random chances of being correctly identified), but it can be used to compare different signals evaluated in the same experiment.

We only considered an annotation to be a hit if its checkbox was clicked after the onset of the anomaly and remained checked until the end of the level. Each "prediction" of an anomaly is considered a false positive and each "change of mind" is counted as no annotation.

See Figure 2 for a histogram of the mean sensitivity index for each stimulus, as well as overall, in each ecology.

- WPD: Droplets is clearly less efficient than Arpeggio. Arpeggio is slightly better in Mixed than Synth.
- PH: Birds is clearly more efficient than Drone.
- WPT: all three ecologies do not present a clear difference, though Water appears to work slightly better in Mixed than in Nature.
- PT: Sizzle is detected much more efficiently in Mixed than in Nature.
- Overall: Mixed is slightly better than Synth, and Nature is visibly less efficient than either.

5.2 Annotation Times

In Figure 3, we plotted the average annotation times for each stimulus type and for each ecology. Once again, these are computed only from cases when participants correctly identified an anomaly after its onset. Comparing those values, we notice that Synth seems to have allowed for better reaction times overall, except for WPD, where the Arpeggio stimulus was annotated faster as part of Mixed than Synth.

5.3 Primary Task Performance

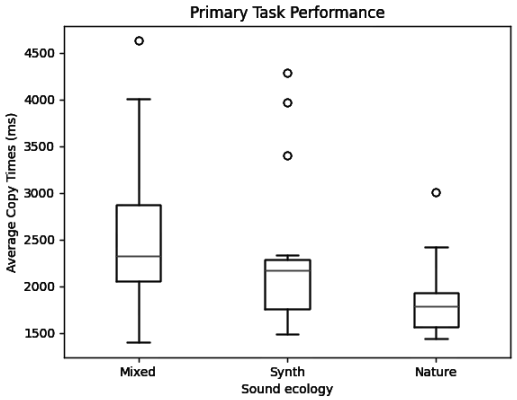


Figure 4. Box-and-whiskers plots of the testers' average sequence-copying time (in ms) for each ecology.

We computed the average time required by each participant to copy a sequence as a metric for their performance at the primary task. By plotting the distribution of those scores as box plots for each ecology (see Figure 4), we get a representation of the overall performance of testers in each condition. In those diagrams, a lower value indicates a better score, since the copying time needs to be as low as possible for an optimal performance.

Nature seems to have the overall best set of participant performances, followed by Synth. The scores are more widely distributed for Mixed, since that experiment got a much larger number of participants. It seems to indicate slightly worse primary task performance than the other two.

5.4 Qualitative Survey

The end survey for the evaluation consisted in a series of seven statements (3 regarding the sound design and 4 regarding the primary task in relation to the sound), to which

	Mixed	Synth	Nature
Easy to distinguish sounds	71.43	82.14	**83.33**
Sounds distract from task	7.14	**25.00**	14.29
Sounds are stressful	19.05	**42.86**	30.00
Task is stressful	16.67	**25.00**	16.67
Task is fun	66.67	46.43	**73.33**
Task is difficult	14.29	**21.43**	13.33
Task distracts from sounds	11.90	**21.43**	3.33

Table 2. Overview of the qualitative survey answers. For each statement, the strongest agreement rates are highlighted in bold font.

participants could respond on a 3-step scale from "Disagree" to "Agree". In order to analyse the answers, we attributed values to each step of that agreement scale, such that "Disagree" was worth 0, "Somewhat Agree" was worth 50, and "Agree" was worth 100. We then averaged those values for each statement, which gave us a collection of percentages indicating the overall agreement rates. See Table 2 for an overview of those answers.

Overall, Synth appears to be considered more stressful than the other two ecologies (rows 2-7). This even includes testers seeing the primary task as more stressful when presented with this set of sounds (row 4).

In a free-form comment for the Synth ecology, one tester reported finding the Drone stimulus too sensitive. That issue had also been reported in the original Mixed evaluation. It seems to indicate that most of the mistakes for Drone may be caused by false alarms rather than missed hits.

The participants who tested Nature seemed to perceive its set of stimuli as easier to distinguish (row 1), despite a worse performance at the annotation task. This may be due to the fact that, although the types of sounds themselves were easy to differentiate, their modulations in case of anomalies may have been more difficult to detect. Two participants reported that they found normal and abnormal situations to be difficult to differentiate for Droplets, which is clearly reflected in its performance and was most likely encountered by more users than just the two who reported the issue.

For Nature, four testers reported having trouble differentiating Sizzle and Water, another issue which had previously been reported for Mixed. This confusion is surprising to an extent as Sizzle was selected to occupy a higher frequency range than Water, and to have a very characteristic transient envelope instead of being a continuous bubbling sound. Although those sounds were supposed to be quite different on a purely perceptual level, it seems that the "everyday listening" may have taken over in peripheral monitoring, and both were simply heard as "the sound of boiling water". Also, since it was a last-resort alarm, Sizzle was more rarely represented than other stimuli in the experiment, which may have led to testers forgetting it after the original exposure during training. We suspect that a proper long-term training may greatly reduce the problems with discriminating Water and Sizzle. Also, we note that these two stimuli still yielded better sensitivity indices in Mixed than any of the other stimuli.

The sounds used in Mixed were rated relatively more often as difficult to differentiate (row 1). This may be due to the combined effects of the aforementioned Sizzle/Water confusion and Drone over-sensitivity.

We notice that the ecologies involving natural sounds tended to be rated more positively:

- Nature stimuli are seen as easier to distinguish (row 1).
- Mixed is the least distracting and the least stressful (rows 2 and 3).
- Mixed and Nature are ex-aequo for least stressful influence on primary task (row 4).
- The primary task is seen as more fun and less difficult with Nature playing (rows 5 and 6).
- Nature is easier to follow while taking care of the primary task (row 7).

These last few points appears to confirm the usual arguments for using auditory icons in monitoring displays.

Finally, three participants for Nature reported that they found the sound of crackling ice used for Water to be either distracting, distressing, or stressful. This issue had not previously been reported for Mixed, and we suspect it may be a new affective side effect of the thematic context for the sounds used in Nature.

6. DISCUSSION

Although our results seem to suggest some trends with regards to our hypothesis, it should be noted that statistical analysis (pairwise one-way ANOVA) indicated that they were not sufficiently representative, so they should be taken with some caution.

The Nature ecology, for which annotation accuracy was the worst, seemed to be more positively received by testers in relation to the primary task, and to allow for a better performance at it. This may be related to a bias in tester demography, as the participants testing Nature were overall younger than those testing Synth (15 of them were aged below 40, against 8 for Synth), so they may just have been more proficient at mouse-based tasks in general. However, such an explanation does not account for the fact that the reaction times were overall shorter for Synth than for Nature. Perhaps testers for Synth and Nature simply prioritized different aspects of the dual task, respectively the annotation of stimuli or the copy of sequences. Still, the qualitative feedback seems to confirm the often-cited preference of listeners for natural-sounding stimuli, which may have transpired into their primary task performance as a sign of a more relaxed monitoring experience.

The fact that Synth was better detected but enjoyed less by testers may be due to the fact that it is closer to the traditional "functional" alarms paradigm, which allows for strong notifications but tend to be more stressful in long-term use.

We notice that some of the stimuli that were shared between ecologies were better detected when presented in the Mixed ecology than in their respective coherent ones: Arpeggio is more efficient in Mixed than Synth, and Sizzle and Water are more efficient in Mixed than Nature. This would suggest a confirmation of our original hypothesis for designing an ecology where different categories of sounds could better stand out from one another.

7. CONCLUSION

As part of our work towards the use of peripheral sonification in 3D printing process monitoring, we have studied the effect of ecological coherence in soundscapes. The comparative evaluation we conducted seems to favour our hypothesis that an incoherent ecology allows for better detection and discrimination of stimuli. Though, we also note that the natural-sounding ecology was seen as less intrusive by users, and that the synthetic-sounding one yielded shorter reaction times. When it comes to those criteria, our Mixed ecology's position between the other two seems to make it a sort of compromise for both low intrusion and fast reaction.

We notice that, although the Nature ecology performed overall worse than the others for stimulus discrimination, Birds did much better than Drone as a stimulus for PH notifications. Assuming incoherent ecology designs are worth researching further, we now wonder if it would be possible to build a better Mixed ecology by including that natural sound instead. More generally, in order to determine as exhaustively as possible which combination of stimuli allows for the best reaction rates, we would probably need to experiment on all of them. This would also allow us to test whether WPT and PT could be detected more accurately if they were conveyed by completely different sound types (Water + Bell, or Jingle + Sizzle).

For this work, we designed sets of stimuli that we deemed representative of coherent ecological considerations for either musical or natural sounds. Although the experiment we conducted seemed to favour the incoherent ecology, we carefully keep in mind that those results are still likely to reflect our own specific design choices rather than the types of sound ecologies themselves. So, we hope that a larger corpus of experiments comparing coherent and incoherent soundscape designs emerges in the future and allows for a more definite conclusion to be drawn on the topic of ecological incongruity in notification systems.

Acknowledgments

The authors of this paper would like to thank the SCRIME of the University of Bordeaux for hosting this research, as well as the Addimadour platform and Anaïs Domergue for allowing us to observe the wire arc printing process in person, and answering our questions on the topic. We would also like to thank Nadine Couture and Sébastien Ibarboure (ESTIA) for initially submitting this case study to us and being greatly involved in the proof-reading and documentation of previous iterations of this work. Finally we want to acknowledge the contribution of Matthias Robine (LaBRI) and Emmanuel Duc (SIGMA Clermont) for their insights into sound design and choice of evaluation process.

8. REFERENCES

[1] G. Kramer, B. N. Walker, T. Bonebright, P. Cook, J. H. Flowers, and N. Miner, "The Sonification Report: Status of the Field and Research Agenda. Report prepared for the National Science Foundation by Members of the International Community for Auditory Display," in *International Community for Auditory Display (ICAD)*, Santa Fe, NM, 1999.

[2] W. W. Gaver, R. B. Smith, and T. O'Shea, "Effective Sounds in Complex Systems: The ARKola Simulation," in *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, 1991, pp. 85–90.

[3] M. Rauterberg and E. Styger, "Positive Effects of Sound Feedback During the Operation of a Plant Simulator," in *International Conference on Human-Computer Interaction*. Springer, 1994, pp. 35–44.

[4] R. D. Patterson, T. F. Mayfield, D. E. Broadbent, A. D. Baddeley, and J. Reason, "Auditory Warning Sounds in the Work Environment," *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, vol. 327, no. 1241, pp. 485–492, 1990.

[5] M. Weiser and J. S. Brown, "The Coming Age of Calm Technology," 1996.

[6] P. Vickers, "Sonification for Process Monitoring," in *The Sonification Handbook*, T. Hermann, A. Hunt, and J. Nuehoff, Eds. Logos Verlag, 2011, ch. 18, pp. 455–491.

[7] T. Hildebrandt, T. Hermann, and S. Rinderle-Ma, "Continuous Sonification Enhances Adequacy of Interactions in Peripheral Process Monitoring," *International Journal of Human-Computer Studies*, vol. 95, pp. 54–65, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S107158191630074X

[8] E. D. Mynatt, M. Back, R. Want, M. Baer, and J. B. Ellis, "Designing Audio Aura," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1998, pp. 566–573.

[9] M. M. Murray, C. Camen, S. L. G. Andino, P. Bovet, and S. Clarke, "Rapid Brain Discrimination of Sounds of Objects," *Journal of Neuroscience*, vol. 26, no. 4, pp. 1293–1302, 2006.

[10] S. G. Lomber and S. Malhotra, "Double Dissociation of 'What' and 'Where' Processing in Auditory Cortex," *Nature neuroscience*, vol. 11, no. 5, pp. 609–616, 2008.

[11] A. M. Leaver and J. P. Rauschecker, "Cortical Representation of Natural Complex Sounds: Effects of Acoustic Features and Auditory Object Category,"

Journal of Neuroscience, vol. 30, no. 22, pp. 7604–7612, 2010.

[12] V. Isnard, M. Taffou, I. Viaud-Delmon, and C. Suied, "Auditory Sketches: Very Sparse Representations of Sounds Are Still Recognizable," *PLOS ONE*, vol. 11, no. 3, pp. 1–15, 03 2016. [Online]. Available: https://doi.org/10.1371/journal.pone.0150313

[13] M. Poret, S. Ibarboure, M. Desainte-Catherine, C. Semal, and N. Couture, "Peripheral Auditory Display for 3D-Printing Process Monitoring," in *Proceedings of the 2021 Sound and Music Computing Conference (SMC 2021)*, Turin, Italy, 2021.

[14] J. Cohen, ""Kirk Here": Using Genre Sounds to Monitor Background Activity," in *INTERACT '93 and CHI '93 Conference Companion on Human Factors in Computing Systems*, ser. CHI '93. New York, NY, USA: Association for Computing Machinery, 1993, p. 63–64. [Online]. Available: https://doi.org/10.1145/259964.260073

[15] M. Gilfix and A. L. Couch, "Peep (The Network Auralizer): Monitoring Your Network with Sound," in *Proceedings of the 14th USENIX Conference on System Administration (LISA'00)*. Berkeley, CA, USA: USENIX Association, 2000, pp. 109–118.

[16] T. Hermann, T. Hildebrandt, P. Langeslag, and S. Rinderle-Ma, "Optimizing Aesthetics and Precision in Sonification for Peripheral Process Monitoring," in *Proceedings of the 21st International Conference for Auditory Display (ICAD 2015)*, Graz, Austria, 2015, pp. 317–318.

[17] S. Lenzi, T. Riccardo, T. Ginevra, S. Galelli, P. Ciuccarelli *et al.*, "Disclosing Cyber-Attacks on Water Distribution Systems. An Experimental Approach to the Sonification of Threats and Anomalous Data," in *25th International Conference on Auditory Display*. International Community on Auditory Display, 2019, pp. 125–132.

[18] A. L. Aldana Blanco, S. Grautoff, and T. Hermann, "ECG Sonification to Support the Diagnosis and Monitoring of Myocardial Infarction," *Journal on Multimodal User Interfaces*, vol. 14, no. 2, pp. 207–218, 2020.

[19] S. Matinfar, T. Hermann, M. Seibold, P. Fürnstahl, M. Farshad, and N. Navab, "Sonification for Process Monitoring in Highly Sensitive Surgical Tasks," in *Proceedings of the Nordic Sound and Music Computing Conference 2019 (Nordic SMC 2019)*, 2019.

[20] S. W. Williams, F. Martina, A. C. Addison, J. Ding, G. Pardal, and P. Colegrove, "Wire + Arc Additive Manufacturing," *Materials Science and Technology*, vol. 32, no. 7, pp. 641–647, 2016. [Online]. Available: https://doi.org/10.1179/1743284715Y.0000000073

[21] S. Ibarboure, "Perception d'Information par des Retours Vibrotactiles pour une Mise en Œuvre plus Flexible des Procédés Robotisés de Fabrication Additive." Theses, Université de Bordeaux, Jun. 2021. [Online]. Available: https://tel.archives-ouvertes.fr/tel-03361862

[22] A. Ceruti, A. Liverani, and T. Bombardi, "Augmented Vision and Interactive Monitoring in 3D Printing Process," in *International Journal on Interactive Design and Manufacturing*, 2017.

[23] M. Poret, S. Ibarboure, M. Robine, E. Duc, N. Couture, M. Desainte-Catherine, and C. Semal, "Sonification for 3D Printing Process Monitoring," in *Proceedings of the 2021 International Computer Music Conference (ICMC 2021)*, Santiago, Chile, 2021.

[24] C. Xia, Z. Pan, J. Polden, H. Li, Y. Xu, S. Chen, and Y. Zhang, "A Review on Wire Arc Additive Manufacturing: Monitoring, Control and a Framework of Automated System," *Journal of Manufacturing Systems*, vol. 57, pp. 31–45, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612520301412

[25] W. W. Gaver, "What in the World Do We Hear? an Ecological Approach to Auditory Event Perception," *Ecological Psychology*, vol. 5, no. 1, pp. 1–29, 1993.

[26] M. M. Blattner, D. A. Sumikawa, and R. M. Greenberg, "Earcons and Icons: Their Structure and Common Design Principles," *Human–Computer Interaction*, vol. 4, no. 1, pp. 11–44, 1989.

[27] W. W. Gaver, "The SonicFinder: An Interface that Uses Auditory Icons," *Human–Computer Interaction*, vol. 4, no. 1, pp. 67–94, 1989.

[28] N. A. Macmillan and C. D. Creelman, *Detection Theory: A User's Guide*. Psychology press, 2004.

Speleoacoustics in Southern Ardèche for Auralizations and Music Experimentation

Luna Valentin
Université Jean Monnet
lunavalentin8599@gmail.com

Miriam Kolar
Stanford University
Center for Computer Research in Music and Acoustics
kolar@ccrma.stanford.edu

Philippe Monteil
CESAME
monteil.philippe@free.fr

ABSTRACT

Caves are archetypically considered to be large-volume and therefore lengthy-reverberating, resonant spaces, and have not been given much consideration in terms of the enormous variety of acoustical environments that they contain. Though cave acoustics have been studied, beyond a recent model of Lascaux we have not seen computational models of cave acoustics as a research focus; for the purpose of relating cave acoustics to human uses of caves, we are engaged in new collaborations to create data-driven acoustical models and auralizations. Here, we propose a human-centered acoustical data collection strategy to enable physics-based and psychoacoustically accurate spatial reconstructions of cave acoustics. These reconstructive models can be used to produce audio demonstrations of cave acoustics in which different sound sources can be auralized. We summarize 2021 speleoacoustics measurements that we made within limestone caves in the Ardèche Valley of south-central France. These measurements reflect methodological propositions for fieldwork relating spatial acoustics to human sensory experience and associated anthropological concerns. We also compare a range of different acoustical features corresponding to specific and contrasting geomorphological contexts in related cave systems. Further, our study was conducted to prepare for future archaeoacoustics research that will offer virtual access to cultural heritage acoustics. These data can be used to produce experiential simulations that create new spaces for musical experimentation.

1. INTRODUCTION

We propose here the term "speleoacoustics" to refer to acoustical studies of caves most generally, whereas "archaeoacoustics" is the name of the field concerned with the study of acoustics in relation with archaeology [1], that is, the study of human life from its materials remains, and therefore a specific interdisciplinary science [2]. Following previous work on Ardèche cave acoustics [3], in conversation with Monteil for expertise on karstic and archaeological background of Ardèche Caves [4, 5] and with Kolar about archaeoacoustical premises, Valentin selected the two caves for the present study according to the following research propositions.

We designed the research discussed here for the purposes of 1) testing the feasibility and in-cave performance of specific audio and documentation equipment for speleoacoustics research; 2) cross-comparing the performance of equipment and measurement signals across a range of cave settings in preparation for a larger research collaboration on the acoustics of caves decorated during the Upper Paleolithic; and 3) collecting and analyzing "human-centered" speleoacoustical data for the purpose of reconstructing cave acoustics in auralizations.

Copyright: © 2022 Luna Valentin et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. DATA COLLECTION METHODOLOGY

2.1 Caves Selected for Fieldwork

Saint-Marcel Cave (La Grotte de Saint-Marcel) and l'Ours Cave (La Grotte de l'Ours – "the cave of bears") are located in the Ardèche Valley of south-central France (in the Municipalities of Vallon-Pont-d'Arc and Bidon, which are 13km apart) which is known in geomorphological terms as a karstic massif of cretaceous limestone drilled during the late Miocene by the Messinian Salinity Crisis [6]. Saint-Marcel Cave covers more than 64km with a vertical drop of 325m and a natural entrance altitude of 95m. l'Ours Cave covers approximately 90m but exists as part of a larger cave system uniting several caves along the valley of the Ibie River (a tributary of the Ardèche River), with an entrance altitude of 200m. The two caves are comparable in terms of their karstic and geological features, but with different geomorphological profiles.

In speleological terms, distinct paths for the passage of ancient rivers are called "galleries". The galleries of Saint-Marcel Cave are very wide (10m to 30m by up to 40m high) with various calcite formations (active or non-active, depending on the location) in some areas. l'Ours Cave is narrower with a main gallery that is approximately 5m wide and 10m high, with non-active calcite formations. Both caves present historical and archaeological use-evidence, with cultural materials previously identified around their historical entrances.

We selected both caves on the basis of their physical and cultural profiles, and also to study features similar to those in the nearby Grotte Chauvet, a venue in which we be-

gan fieldwork in March 2022. The relative accessibility of the selected caves was important to our purpose of initiating joint fieldwork between archaeoacoustician Kolar and spelunker Valentin, with the support of local caves expert Philippe Monteil. Saint-Marcel and l'Ours are both referenced as "Grade 1 caves" (they do not require the use of any spelunking equipment other than a helmet with lighting) by the French Federation of Spelunking. Both caves in which we conducted this fieldwork have evidence of past human uses; however, our purpose here was not the integration of our speleoacoustical research with archaeological data, but rather, the documentation of particular cave features in terms of a human-centered data collection method. This work produces spatially accurate documentation of human perspectives on a specific soundfield context, enabling realistic auralizations over headphones [7].

2.2 Equipment for Speleoacoustical Measurements

One goal of this fieldwork was in-cave testing of the performance of audio equipment and source test signals. Speleoacoustics research is logistically constrained by the location and conditions of caves. First, and as with spelunking, the portability and durability of the gear necessary for cave activities is one of the main points to consider. Even if the accessibility of the caves is ensured, transporting the gear to rural cave entrances and throughout cave environments presents challenges in terms of form-factor, weight, and number of bags. Second, gear must be self-powered, which reduces the selection of equipment for both sound sources (loudspeakers and impulse-generation tools) and receivers (microphones, preamplifiers, and audio recorders, as well as recording devices, including audio interfaces and computers). Third, climate conditions had to be monitored to observe equipment reliability in cave conditions, as well as produce documentation to track the speed-of-sound for certain analytical techniques. Thanks to the Amprobe C02 meter, we were able to track temperature, humidity and the C02 levels that can affect human health and safety in limestone caves, as listed in Table 1.

Amprobe	Temperature [°C]	Humidity [%]	C02 [ppm]
Saint-Marcel	12.1 to 14	88 to 99.9	462 to 856
l'Ours Cave	12 to 12.5	91 to 98	552 to 575

Table 1. Climate conditions measured in the study caves.

We used three setups for impulse response (IR) measurements, the standard technique for documenting the acoustics of enclosed spaces. 1) For synchronous playback and recording of audio signal sources, our most complicated setup employed a Meyer MM-4XP precision loudspeaker and a JBL BassPro SL 8" compact subwoofer, both powered by the Jackery Explorer 240W Portable Power Station though used sequentially in playing test signals. The measuring laptop computer used as its USB-audio interface a Zoom F8N field recorder, running Digital Peformer 10 software both to play the audio signal of a 40-second all-pass chirp that we had generated in Matlab from code by Jonathan Abel, and also to record the "room" responses ("synchronous playback and recording"). Post-processing was then necessary to convert these raw response recordings into impulse responses, again using a Matlab script by Abel. 2) For asynchronous measurements (where playback and recording equipment is not interconnected), we played the same 40-second all-pass chirp from an iPhone plugged into a portable omnidirectional loudspeaker, the Bose Soundlink Revolve Plus II. The raw responses from these asynchronous measurements were likewise processed into IRs using a customized Matlab script by Abel. 3) For the most portable impulse generation system, we carried with us two pairs of wooden clappers. For receivers, we used through Zoom recorder preamplifiers one pair of AuSIM in-ear omnidirectional microphones (Sennheiser KE4-211-4 capsultes) and two Behringer ECM-8000 omnidirectional measurement microphones. Additionally, we used two first-order Ambisonics (FOA) microphone arrays (set to record in 4-channel A-format) via Zoom H3VR field recorders that we configured in the custom setup discussed below.

Following Kolar's prior archaeoacoustics fieldwork approach [8], audio source and receiver positions used in recording impulse response measurements serve as proxies for human sound-generation and auditory reception. Loudspeakers (except the ground-located subwoofer) were located in correspondence with human vocal production, at an approximate head-height (1.5m) at the level of the ear-height of the researcher wearing the in-ear (blocked meatus) binuaral microphones. The microphone arrays of analytical interest to our discussion here, two portable first-order Ambisonics (FOA) recorders (Zoom H3VR) were arranged to locate their "left-up" (channel 1) capsules as spatial proxies for in-ear binaural microphones, spaced 17cm apart in accordance with the typical interaural distance used by the ORTF coincident stereophony standard. This double-FOA "binaural" setup was proposed by Kolar for its spatial sampling accuracy as a proxy for human ear locations (head and torso/body features can be then estimated with filters), then augmented with height channels to produce the "W-Ambisonics" 3D-microphone recording technique developed and perceptually tested by Lu and Kim for multi-speaker spatial audio rendering [9].

2.3 Measurement Strategies and Locations

We chose measurement locations according to comparable and distinct physical features to collect data for quantifying and demonstrating how different geomorphological features relate to cave acoustics. In our comparison, we present the following cave contexts:

In Saint-Marcel Cave, we studied three specific locations in the same extensive gallery (known as Web 1). Two of these are located near the cave's natural entrance, while one is located in a gallery called "Les Boas". The two final locations of our study are situated in l'Ours Cave.

Location (A): Saint-Marcel Cave, "Les Boas" Gallery (Fig.1). This wide gallery is considered the archetype of all Saint-Marcel Galleries, surrounded with smooth limestone shapes. At 20m by 10m, Les Boas ("the boas") is

Figure 1. Equipment setup in "Les Boas" Gallery of Saint-Marcel Cave (Location A).

Figure 2. Equipment setup in the main gallery of l'Ours Cave (Location B).

named after the snake-like calcite formations on its ground: sinter-pool or rimstone pool formations that run all over the gallery. Between those calcite formations, the ground is covered with clay. We located our audio equipment to make measurements somewhat equidistant from walls or pool barriers, centering both sources and receivers near the northern end yet within its large open volume.

Location (B): l'Ours Cave, main gallery (Fig.2). This gallery is a small volume, 10m by 7m, with a smooth limestone profile and ground covered with clay. We positioned both the sources and receivers about 50cm from the south wall of the gallery, in a line approximately parallel to that wall.

Location (C): l'Ours Cave, cavity (Fig.3). In this niche-like location adjacent to the main gallery in the l'Ours Cave, we positioned both sources and receivers along the center of a narrow cavity with a diameter of approximately 5m, surrounded by a great concentration of calcite speleothems.

Location (D): A single-domed ceiling area in Saint-Marcel Cave. We positioned both sources and receivers centered under this 2.5m x 2m x 1m dome.

Location (E): A multi-domed ceiling area in Saint-Marcel Cave (Fig.4). We selected this final testing location of the study due to its profile similarities with the so-called "Woman-Bison" feature of nearby Chauvet Cave, with a smooth and pocketed limestone ceiling.

Figure 3. Equipment setup in the speleothem-lined cavity of l'Ours Cave (Location C).

Figure 4. Equipment setup under the multi-domed ceiling of Saint-Marcel (Location E).

3. DATA ANALYSIS AND INTERPRETATION

Prior studies of cave acoustics have been predicated on hearing echoes and noting tonal responses to the human voice, particularly in contexts of parietal art [10]. Consequently, acoustical research on caves has been preoccupied with exploring the relationship between wall paintings and notable sound effects [11] [12]. We assert there are additional sonic factors of interest. The topic of acoustical parameterization in archaeocoustics has attracted controversy since that field's earliest organized professional event at the McDonald Institute for Archaeological Research at Cambridge University, documented in the edited volume "Archaeoacoustics" by Scarre and Lawson [13]. Relating measured acoustical metrics that have been developed for documentation of the recent built environment presents considerable challenges when applied in ancient and landform contexts such as caves. Ian Cross and Aaron Watson explored this problem in the aforementioned volume, questioning, "can we use this information represented in these measurements to understand the substance of social activities involving sound in cultures and periods other than our own?" [13]:107.

Some acoustical parameters might be more relevant in cave studies than others. Here, we address the question of analytical translation across time and human contexts in terms of key metrics from the international room acoustics measurement standard ISO 3382, with the premise that

cave environments are enclosed spaces whose sonic prolongation and coloration features relate to commonly noted perceptions of caves, and with the intention of relating our study to acoustics literature on caves which uses these metrics. In addition to the ubiquitous RT30 metric, we bring to attention the Early Decay Time (EDT) metric because it describes the initial acoustical energy that provides important perceptual context cues about human's immediate surroundings [14]: 27-30. For this reason, the measurements in this study were taken within the critical distance between source and receiver, in which the direct sound energy is higher than the reverberant (indirect) energy. In more comprehensive speleoacoustical surveys, we would produce measurements reflecting a greater range of distances between source and receivers.

We display our impulse response analyses using the commercial acoustical measurement software RØDE Fuzzmeasure; however, during our analyses, we also verified values using the shareware tool Room EQ Wizard, and we checked some band calculations in other software. Because each day of our 3-day case-study tested a specific combination of methodological propositions — using different equipment and configurations per day — we do not present here an equal comparison across all cave settings. Rather, we use comparative techniques to highlight observations about equipment performance, as well as our purpose in demonstrating the great variety of speleoacoustical features present in just two cave systems within the same region.

3.1 Spatial Acoustics, Equipment Performance, and Acoustical Metrics: A Web of Possibilities

We examine here the cave impulse response data collected using both mathematically generated acoustical test signals via loudspeakers and human-produced impulsive sounds. Note that we did not bandlimit the full-range test signal played through the subwoofer; therefore, its production of mid and high frequencies should be excluded from the analysis, but we graph its output to document its performance, which our study tested.

First, in a central location with a dry clay floor within a large-volume cave (Les Boas, Saint-Marcel; Location A; Fig.1) we compare the following impulse response measurement sources, recorded with an omnidirectional microphone (Behringer ECM-8000): a) a human-produced broad-band impulse (via wooden clappers); b) broad-band loudspeaker-reproduced sinesweeps through a Meyer MM-4XP single-driver loudspeaker (120Hz–18kHz) positioned at 1.5m above the ground; c) the same sinusoidal signals through a JBL BassPro SL8 subwoofer (35-120 Hz), located on the ground under the Meyer radiating in an upward hemisphere.

This experimental construct enables us to quantify the spatial response differences in produced acoustical energy between a human-produced impulse and loudspeakers with different spectral profiles. We selected the loudspeakers on the basis of their quality vs. portability features including size and power consumption. We note that the directional Meyer MM-4XP has been used extensively as a proxy for vocal and aerophone sources in our previous archaeoacoustics fieldwork [15].

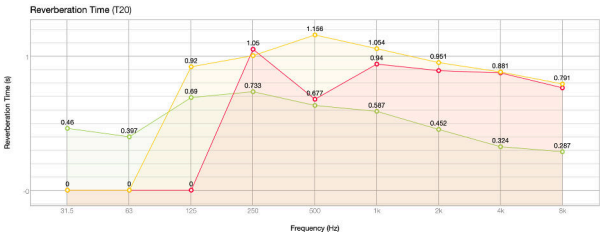


Figure 5. RT20 in two caves calculated from human-produced impulse responses from wooden clappers in Saint-Marcel (yellow & red) and l'Ours (green).

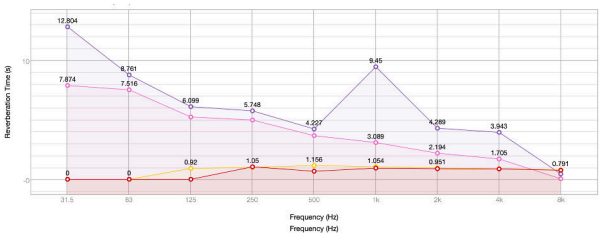


Figure 6. RT20 comparison among the sources in Les Boas gallery of Saint-Marcel cave: 2 claps (yellow & red), Meyer-MM4XP miniature loudspeaker (pink), and JBL BassPro SL8 compact subwoofer (purple). Note the excellent agreement among loudspeaker sources yet with spurious values above 500 Hz for the subwoofer, which distorts higher frequencies.

We compare in Fig.10 the above impulse response measurements to those made with the same equipment in the main central room within a small-volume cave (l'Ours; Location B, Fig.2). In l'Ours, the measurements were adjacent to a smooth, slightly pitted and somewhat curvy limestone wall, with a damp clay floor.

3.1.1 IR Measurement Analysis in Saint-Marcel Les Boas / Location (A)

Human-produced impulsive signals such as those from our wooden clappers can be particularly useful in speleoacoustics because of their portability; however, they are not reliably full-spectrum. We used them with caveats, and here evaluate their performance in different cave settings. One of the functional issues is the produced energy across the frequency spectrum, whose requirement varies depending on the dimensions of the space being measured. Comparing two human-clapped impulses using the same wooden clappers, in the same location, produced slightly different estimations of reverberation time (claps, RT20, in Fig.5, in yellow and red). We noticed a lack of low-frequency energy, and can attribute it to the source production through comparison using IRs created via a 20Hz-20kHz, 40-second sinusoidal sweep through both a subwoofer and a precision miniature loudspeaker (Fig.6). The RT20 comparison of these 3 test signal sources in Les Boas tracks well into plausible values in the lowest frequencies, around 12 seconds in the 31.5Hz-centered band, reducing

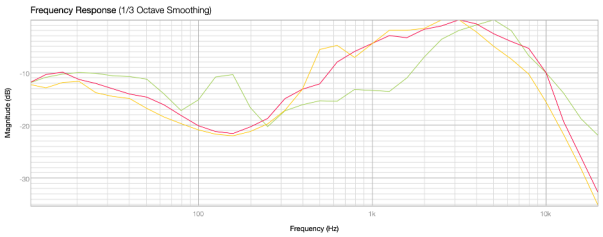


Figure 7. Normalized magnitude frequency response of three human-produced IRs with third-octave smoothing: two from Les Boas (red & yellow) and one from the much smaller l'Ours cave's main gallery (green).

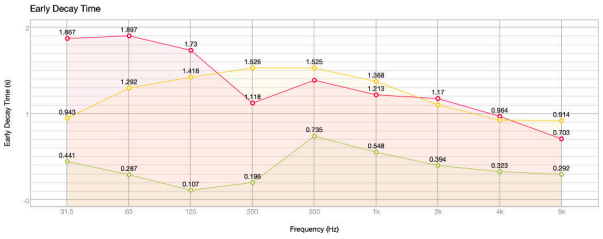


Figure 8. EDT values from two caves, calculated from human-produced impulse responses from wooden clappers in Saint-Marcel (yellow & red) and l'Ours (green).

to 4-6 seconds in the mid-range, and about half that around 1kHz (note that the 9-sec value at 1kHz for the subwoofer has been miscalculated by the software, due to distorted production in this range). For mechanical comparison, we checked the same clapper source from our measurements in l'Ours Cave, which demonstrates that the clapper is capable of producing low frequency energy (Fig.5, in green). The normalized magnitude frequency response graph of all three clapped IRs demonstrates good correspondence across the spectrum (Fig.7).

Another analytical tool for understanding the energetic profile of sound sources within the critical distance (where direct energy is higher than reverberant) is the EDT metric. For our clapped IRs in Les Boas, EDT values in low frequency bands are greater than for the RT20 metric, which corroborates our interpretation that produced low-frequency energy was very low for the clap-generated IRs in proportion to the spatial dimensions to be engergized (Fig.8). In comparing the non-normalized magnitude frequency response of the three sound source types in Les Boas (Fig.9), we verify the consistency between these two clap-produced IRs, as well as the relative greater energy in low frequencies of signals produced by the JBL subwoofer, and also the broadband regularity of the Meyer loudspeaker.

3.1.2 IR Measurement Analysis in L'Ours Cave Locations (B) and (C)

In both the compact cave system of l'Ours and in the enormous gallery of Saint-Marcel Les Boas, we used the same equipment and procedures to make acoustical measurements using both loudspeakers and human-generated impulses. (Fig.10), a graph of RT20, shows excellent agree-

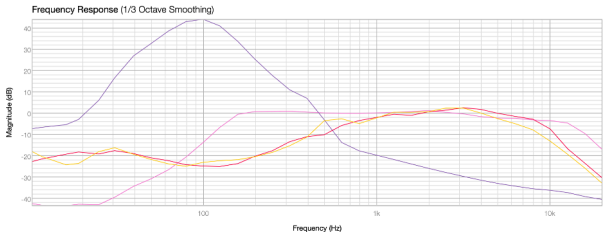


Figure 9. Les Boas sound sources – non-normalized, magnitude frequency response of the three source types, with third-octave smoothing: 2 human-produced impulses with wooden clapper (yellow & red), Meyer-MM4XP miniature loudspeaker (pink), JBL BassPro SL8 compact subwoofer (purple).

ment across sound-production methods. Based on this demonstration, and from cross-comparison across different RT calculation methods, we believe that the 4-second reverberation time in the lowest octave band is correct, as well as the generally decreasing RT/frequency towards the figure of 0.28 seconds in the 8kHz-centered band. Further, our personal perceptions of the relatively longer low-frequency reverberation in both caves is corroborated by these data. In (Fig.11) we present the magnitude frequency response of the clapped impulse (green), miniature loudspeaker (pink), and subwoofer (purple) in order to show that l'Ours cave's geomorphological profile has particular implications for high-frequency sound. Observe the trend in both clapped IR and Meyer-speaker-produced IR: an irregular and decreasing frequency response starting around 400Hz and rolling off rapidly with increasing frequency. Due to the relatively low volume of the cave, we hypothesize that this high-frequency absorptiveness is not predominantly due to distance-based air absorption – as might be the case in a voluminous cave, such as Saint-Marcel – but rather due to the many air spaces within the small-scale irregularities of its stone surfaces.

It is informative to compare two areas in the main room of l'Ours Cave, via RT30 (Fig.12) and EDT (Fig.13) values computed from clapped impulse responses recorded on only one cardioid microphone of our H3VR Ambisonics recorders. These data demonstrate the consistency in overall frequency characteristics of this room because of its relatively small size, whether the microphone receivers were adjacent to a smooth limestone wall (the red line) or closer to the center of that volume (the blue line). One of the striking features is the 50Hz room mode that is visible in the RT30 curve, because it appears as a boost in one position and a notch in the other. Similarly, there is clear modal activity at 125Hz. Both positions' high-frequency response curves are nearly identical, and we see the same trend when comparing the EDT (Fig.14) and RT30 (Fig.15) of measurements made within the adjacent speleothem-lined cavity. Of importance here is the extreme difference in frequency profiles between the early reverberation and its later development. Likely due to the rapid interactions of high frequencies with cave structures such as small cavities with calcite formations, the early re-

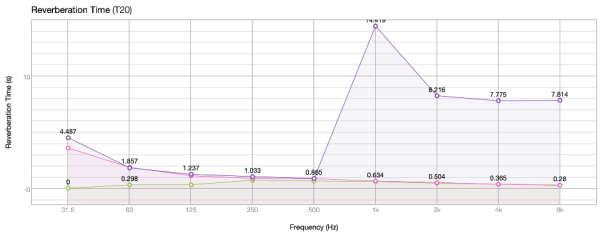


Figure 10. RT20 values from the three sound sources in l'Ours Cave's main gallery: human-produced impulse from wooden clapper (green), Meyer MM-4XP miniature loudspeaker (pink), JBL BassPro SL8 compact subwoofer (purple). Note erroneous values for the subwoofer from the distorted signal above 500Hz.

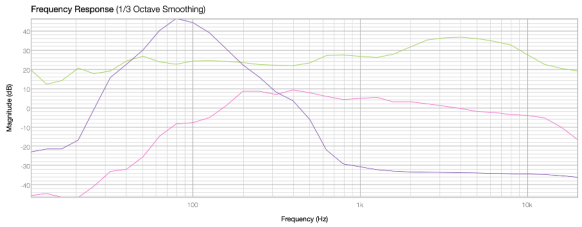


Figure 11. l'Ours sound sources – non-normalized, magnitude frequency response of the three source types, with third-octave smoothing: wooden clapper (green), Meyer MM-4XP miniature loudspeaker (pink), JBL BassPro SL8 compact subwoofer (purple).

verberation as quantified by the EDT metric is dominated by mid-frequencies, whereas over time, higher frequencies disappear and lower frequencies are prolonged according to the larger dimensions of the cave. Without attention to the time-dependencies of reverberation metrics, this particular behavior could be overlooked, with psychoacoustical implications.

3.2 "Human-Centered" Comparison of Speleoacoustical Features

The above contrasting features in cave measurement scenarios are comparable due to consistency of equipment and procedure. Fundamental to our research is the proposition that speleoacoustical measurements are best related to human experience when conducted in terms of human spatial scaling. In other words, acoustical test signals are produced where humans could make sound, with microphone receivers located in humanly plausible locations to enable spatially accurate translations of measured acoustics into auralizations and data-driven models.

Towards this goal, we cave-tested a time-code synchronized pair of Zoom H3VR first-order Ambisonics microphones (the double-FOA "binaural" setup or "double-H3VR array") in which we take the signal from only the "left-up" (channel 1) of each H3VR, positioned according to a 17-cm interaural distance to approximate binaural hearing spatial sampling, with those capsules oriented outwards as proxies for in-ear microphones. These data can

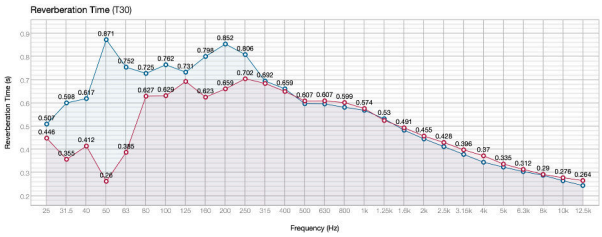


Figure 12. RT30 calculated from clapped impulse responses measured at two locations within l'Ours cave's main gallery: one at the first position, by the wall (red), and a second position more central to the larger volume, farther away from the highly variable surface of the limestone walls (blue).

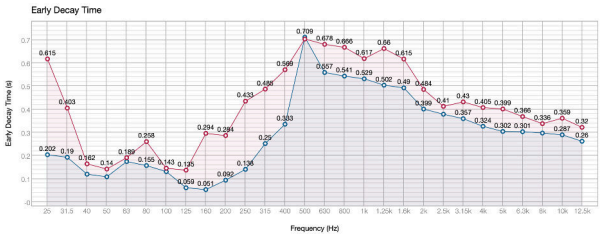


Figure 13. EDT calculated from clapped impulse responses measured at two locations within l'Ours cave's main gallery: as above in Fig.12.

be analyzed directly to estimate the differences in soundfield reception at each ear of a human listener, and both the 2-channel reduction and the 8-channel double-FOA data can be applied in the rendering of immersive binural auralizations. Via the following RT graphs, (Fig.16) and (Fig.17), we offer an overview of the great variety of speleoacoustical features with quantifiable binaural consequences that can be observed and measured even within the same gallery of a single cave system, or within geomorphologically similar caves. We have targeted specific features within the southern gallery of the large Saint-Marcel cave system, and explored some of those features in comparison with measurements from the smaller l'Ours cave.

3.3 Analytical Considerations and Future Directions

A concise preliminary study, such as we present here, that compares equipment across select locations in two caves cannot thoroughly cover all methodological premises for speleoacoustics. Of interest in our future research is the development of strategies for comprehensively exploring, documenting, and evaluating the acoustics of an entire gallery or the complex interconnectivities among spaces within an extensive cave system, such as the research we have begun in Grotte Chauvet fieldwork in March 2022. It is apparent from our initial analysis of measurements from a relatively tiny area within the voluminous space of Les Boas (which covers only about 100m within an interconnected cave system that extends for several kilometers, with large cross-sectional dimensions) that measured RT has particularly local constraints according to source fea-

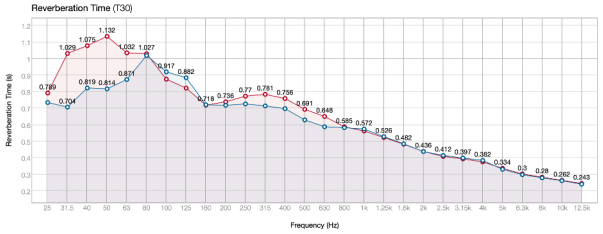


Figure 14. RT30 calculated from clapped impulse responses measured at the two, binauarally spaced outward-facing "left-up" channels of the double-H3VR array in the speleothem-lined cavity of the l'Ours cave.

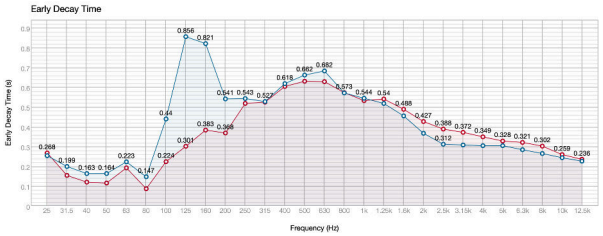


Figure 15. EDT calculated from clapped impulse responses measured as above in Fig.14.

tures and particularly produced sound energy – as well as the gain structure and sampling resolution of the recording devices – in response to the physical scale of the cave setting.

Through selected examples of spatial impulse response measurements made according to locations of plausible human activity, and using human sound-producing and binaural-receiving proxy locations for sources and receivers, we have demonstrated an enormous range of acoustical contrasts that correspond with a diversity of cave features. By comparing the measurable effects of different source production and receiver locations, we show how acoustical measurement results are contingent upon contextual factors that complicate the application of standard acoustical test signals, procedures, and metrics.

4. SPELEOACOUSTICS APPLIED: AURALIZATIONS FOR MUSIC EXPERIMENTATION

We emphasize that this speleoacoustics study is a preliminary exploration to inform our future acoustical studies of Ardèche limestone caves, together and in collaboration with a cross-disciplinary team with an archaeoacoustical premise. We discuss here some considerations in building strong methodological strategies to guide us in producing accurate and extensible acoustical data with relevance to human experience in caves. Preliminary findings on the acoustics of these caves also inform rationales for new strategies to "human-center" speleoacoustical data collection. Our fieldwork here produced the first of a growing collection of speleoacoustical data to be applied in auralizations and immersive experiences, for archaeological research, music experimentation, and novel applications yet

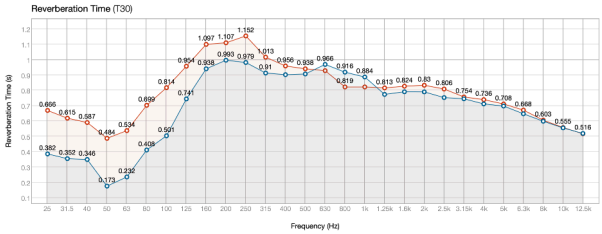


Figure 16. RT30 calculated from clapped impulse responses measured at the two, binauarally spaced outward-facing "left-up" channels of the double-H3VR array located underneath the Saint-Marcel multi-domed area pictured in Fig.4.

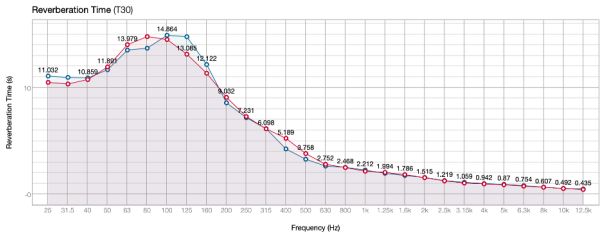


Figure 17. RT30 calculated from clapped impulse responses measured at the two, binauarally spaced outward-facing "left-up" channels of the double-H3VR array located under a small ceiling dome in a wide gallery area in the Saint-Marcel cave.

to be developed. Also, it is worth noting that detailed acoustical parameters of caves would be useful in developing realistic virtual reality models that are representative of actual caves and specific cave features.

Multi-sensory models of caves are providing new forms of access to cultural heritage spaces and their scientific explorations. Vectors for many scientific fields such as geology, hydrology, karstology, climatology, and zoology, caves are exciting places for popularizing science: for example, the French Federation of Spelunking is bringing new visitors to caves through touristic access and spelunking activities that relate to scientific knowledge. But caves are also spaces for scientific explorations of cultural heritage, increasing knowledge of human life across time via archaeology, which is visible in France in particular, regarding Paleolithic use of caves for ritual as well as more recent activites that left traces in Ardèche caves [4, 5]. To enable public access to Paleolithic cultural heritage, caves secured for conservation such as the well-known Lascaux Cave and Chauvet Cave – and the undersea Cosquer Cave, whose replica opens in June 2022 – have been partially reconstructed at scale, with masterful reproductions of their parietal art. The Chauvet and Lascaux replica caves are among the 100 most-visited tourist attractions in France, bringing visitors from around the world, and virtual visual reconstruction platforms are increasing in popularity, without attention to accurate sonic simulations. Following the idea that caves offer new opportunities for popularization and transmission of scientific and cultural knowledge, replicas and virtual models of caves would benefit from

immersive, physically accurate and dynamic sonification.

Although the research area relating cave acoustics with archaeology was many years ago initiated by Reznikoff and Dauvois, recent advances in archaeoacoustics, applied responsively in cave research, can produce new forms of acoustical models and auralizations relevant to human perspectives within caves. The next step is to produce spatially contextualized reconstructions of Paleolithic music, informed by anthropological science in collaboration with skilled musicians. Indeed, we can build imaginative responses to the extensive traces of musical instruments preserved since the Paleolithic; many studies have been conducted to assemble clues and build inferences about musical practices during Paleolithic times [16]. Research on music archaeology supported by archaeoacoustics enables physically accurate virtual explorations of Paleolithic music in relationship with Paleolithic visual expressive culture. Carefully documented impulse response measurements from speleoacoustical fieldwork as discussed here enable accurate data-driven auralizations of real caves, grounding virtual explorations of Paleolithic sound environments.

Acknowledgments

We thank Chris Chafe, Director of the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University for startup funding of the project organized by composer John Chowning to collaborate with archaeologist Carole Fritz and the Grotte Chauvet Research team she leads. Thanks to CESAME for spelunking gear loan and logistics help (Bertrand Hamm); to Delphine Dupuy for access to Saint-Marcel Cave. Luna Valentin's Master's Thesis (2022; Université Jean Monnet de Saint-Etienne, directed by Laurent Pottier and Romain Michon) grounded preliminary fieldwork and questioning on cave acoustics that we applied in our analyses. At the time of our fieldwork, Miriam Kolar was funded in part by the National Endowment for the Humanities in the U.S.A. for her role as co-organizing researcher in the project for Digital Preservation and Access to Aural Heritage Via A Scalable, Extensible Method: `www.AuralHeritage.org`.

5. REFERENCES

[1] C. Scarre and G. Lawson, *Archaeoacoustics*. University of Cambridge, 2006.

[2] M. Kolar, "Archaeoacoustics: Re-sounding material culture," in *Acoustics Today*, 2018, pp. 28–37.

[3] L. Valentin and P. Monteil, "Mesures acoustiques dans la grotte du Déroc (Vallon Pont d'Arc - 07)," 2022.

[4] P. Monteil and O. Peyronel, "Les précurseurs de la spéléologie à Vallon-Pont-d'Arc," in *Spelunca N°141*, march 2016, pp. 13–18.

[5] P. Monteil and O.Peyronel, "L'exploration spéléologique dans la première moitié du xxème sciècle à Vallon-Pont-d'Arc," in *Spelunca N°149*, march 2018, pp. 50–57.

[6] L. Mocochain, J.-Y. Bigot, G. Clauzon, M. Faverjon, and P. Brunet, "La grotte de Saint-Marcel (Ardèche) : un référentiel pour l'évolution des endokarsts méditerranéens depuis 6 ma," in *Karstologia : revue de karstologie et de spéléologie physique*, https://doi.org/10.3406/karst.2006.2587, n°48, 2006, pp. 33–50.

[7] M. Kolar, D. Ko, and S. Kim, "Preserving human perspectives in cultural heritage acoustics: Distance cues and proxemics in aural heritage fieldwork," in *Acoustics*, 2021, pp. 156–176.

[8] M. Kolar, A. Goh, E. Gálvez-Arango, B. Morris, A. Romano, S. Turley, S. Colello, W. Penniman, J. Boffa, C. Wang, G. DePaul, and K. Keene, "Archaeoacoustics fieldwork for aural heritage conservation: Collaborative distributed sound-sensing at Chavín de Huántar, Perú," in *Change Over Time*, 2019, pp. 164–191.

[9] X. Lu, S. Kim, M. Kolar, and D. Ko, "Perceptual evaluation of a new, portable three-dimensional recording technique: 'W-Ambisonics'," in *AES Engineering Briefs 652*, 2021.

[10] I. Reznikoff, "Sur la dimension sonore des grottes à peinture du paléolithique," in *Compte Rendu de l'Académie des Sciences de Paris, tome 304, Série II, n°3*, Paris, 1987, p. 307.

[11] B. Fazenda, C. Scarre, R. Till, R. J. Pasalodos, M. R. Guerra, C. Tejedor, R. O. Peredo, A. Watson, S. Wyatt, C. G. Benito, H. Drinkall, and F. Foulds, "Cave acoustics in prehistory: Exploring the association of Palaeolithic visual motifs and acoustic response," in *The Journal of the Acoustical Society of America*, 2017, pp. 1332–1349.

[12] D. Commins, Y. Coppens, and T. Hidaka, "Acoustics of the Lascaux cave and its facsimile Lascaux IV," in *The Journal of the Acoustical Society of America*, 2020, pp. 918–924.

[13] I. Cross and A. Watson, "Acoustics and the human experience of socially-organized sound," in *Archaeoacoustics*, 2006, pp. 107–116.

[14] D. B. E. Wenzel and M. Godfroy-Cooper, "Perception of spatial sound," in *Immersive Sound: The Art and Science of Binaural and Multi-Channel Audio*, 2018, pp. 5–39.

[15] M. Kolar, J. Rick, P. Cook, and J. Abel, "Ancient pututus contextualized: Integrative archaeoacoustics at Chavín de Huántar, Perú," in *Flower World – Music Archaeology of the Americas, Vol. 1*, 2012, pp. 23–53.

[16] C. Fritz, G. Tosello, G. Fleury, E. Kasarhérou, P. Walter, F. Duranthon, P. Gaillard, and J. Tardieu, "First record of the sound produced by the oldest Upper Paleolithic seashell horn," in *Science Advances*, February 2021.

SMC-22 Paper Session 2

How Live is Live Coding? The case of Tidal's Longest Night.

Raphaël Maurice Forment
ECLLA, Université Jean Monnet (UJM) - Saint Étienne
raphael.forment@gmail.com

Alex McLean
Then Try This, Sheffield/Penryn
alex@slab.org

ABSTRACT

From the 21st to the 23rd of December 2021, more than one hundred live coders from the TidalCycles [1] community and beyond coordinated to produce non-stop algorithmic audiovisual performances broadcasted openly on streaming platforms [1] . For the analyst or software designer, this event represents an extraordinary opportunity in that it allows to study a wide and diverse range of performers incorporating live coding techniques in their work from around the world. In this article, we present a discussion about the particular *liveness* [2] of live coding based on our observations of these performances. We focus on code editing practices and on the general approach to the medium – the text editor and the source code – to better understand how live coders from the event approached the performing act and how they designed live performances through the mediation of code. More specifically, this article explores the reasons that could explain why, despite the fact that live coding libraries such as TidalCycles are built to encourage free improvisation and free exploration of musical patterns, the majority of live coders do not start their performance *from scratch*.

1. INTRODUCTION

The practice of live coding in musical and audiovisual performance is now an established part of computer music research. More than two decades of research and artistic creations have lead to popularisation of live coding techniques among a larger audience thanks to robust audio coding platforms such as SuperCollider [3], Max, Pure Data, ChucK [4], ExTempore [5] and to various musical libraries or software exploring different models of live computation (ORCA [2] , Sonic Pi [6], Gibber [7], TidalCycles, etc...). This popularity emerged from the founding of the TOPLAP collective in 2004 [8], and subsequent initiatives such as the live.code.festival in Karlsruhe, /* vivo */ festivals in Mexico City [9], and Algorave algorithmic dance music events founded in London and quickly spreading across hundreds of cities and festivals worldwide [10].

Musical live coding can now be found in academia [11] as a tool for experiments on new instruments, digital art, computer languages, HCI and pedagogy among many fields of musical, graphical and choreographic research. It has also garnered the interest of enthusiast electronic musicians willing to rethink the role of computers in modern music-making setups, searching for ways to complement – or supplement – complex hardware and/or digital audio workstations. Beyond the numerous possibilities of use, live coding is also perceived by its community as a codified but flexible approach to electronic-music making promoting open-source initiatives, collaboration and values usually associated with the *hacker* ethic and FOSS (Free/Open Source Software) communities [12].

The event we hereby study is by no means unusual for its duration and scale in the live coding world. Indeed, the TOPLAP 15th anniversary (February 19th 2019 [3]) or New Moon (August 19th 2020 [4]) could have provided similar corpuses with even more performances to analyse. This could easily be explained by the fact that having deep roots in *net-art*, creative coding and underground computer music scenes, live coding has always used the internet as the medium *by default* for exchanging knowledge, ideas, software and to create digital meeting grounds. Years before the systemic shift caused by the COVID-19 pandemic and its impact on live music performances, live coding related research already held interest for the topic of network collaboration and synchronisation at its core: Troop [13], Flok [5] , Estuary [14].

In this article, we draw on the mass of performances from the Longest Night stream, in order to nourish a discussion about the presentation of *liveness* in live coding practices and about how different approaches to the medium lead to very different artistic outcomes and performance practices. While all live coders share a common appeal for *liveness* (as far as we know there were no pre-recorded performances during the stream), how the ultimate form in which coding takes place on stage can dramatically shift, from complex pre-written performances to from-scratch improvisation of sound synthesis algorithms.

[1] https://night.tidalcycles.org/: event website.

[2] https://github.com/hundredrabbits/Orca: By David Mondou-Labbe, an *esoteric programming language designed to quickly create procedural sequencers*, following an unusual bi-directional programming paradigm similar to the Befunge programming language.

[3] https://toplap.org/toplap-15th-anniversary-stream-14-17th-february-2019/: event link.

[4] https://sun.tidalcycles.org/: event website.

[5] https://github.com/munshkr/flok: *Web-based P2P collaborative editor for live coding music and graphics*, by Damián Silvani.

Copyright: © 2022 Raphaël Maurice Forment et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. EVENT DESCRIPTION

Figure 1. Longest Night flyer, made by Luluganeta, in charge of the graphical chart.

The *Longest Night* was so called because it took place over the weekend of the December solstice, the moment when the earth finishes one oscillation and begins another. The focus on the solstice is therefore intended to unify an international line-up of people contributing performances (although of course in the Southern hemisphere, this was the *shortest* night). This was a volunteer-run event, organised by members of the international TidalCycles community: Bernard Gray (*Cleary*), Thomas Grund (*MrReason*), Scott Fradkin (*Sfradkin*), Renzo Torrisi (*Ritchse*) and many others active under nicknames: *paulchannelstrip*, *luluganeta*, *Givo29*, *kit-christopher*, *arethusa*, etc... The event emanated from a community with close ties to the software, almost all of the performances were making at least some use of Tidal [6]. Nonetheless, some performers have chosen to use other live coding environments/software such as Gibber, Sonic Pi, ORCA or even, in some cases, custom-made tools and libraries.

The constraints linked to the organisation of an international remote event were handled by leveraging several web technologies, the most important one being *muxy* [7], a server-side tool for multiplexing video streams developed by Tidal community member Damián Silvani, with a web front-end developed for the event by Thomas Grund. A great deal of preparatory work was done to automate the inscription and distribution of stream tokens across multiple time zones, allowing a performer to broadcast and perform during a strict 20 minute slot, before an automatic relay is established with the next performer. There was no vetting of performances, anyone could freely sign up for a performance slot. Technical monitoring and archive recording was ensured for the total duration of the event. Documents and details related to the organisation and technical management were compiled and stored by Bernard Gray to ensure the later reproducibility of the event [8].

With the organisation team taking care of such technicalities, performers were left with the task to connect using audiovisual streaming software and were interacting with each other through various chat systems: YouTube, Discord, Telegram, Twitch or personal communications on social media. Communication could also happen directly through code, the artists using code comments as a way to communicate while performing or even, in some cases, chats directly integrated in the editing environment.

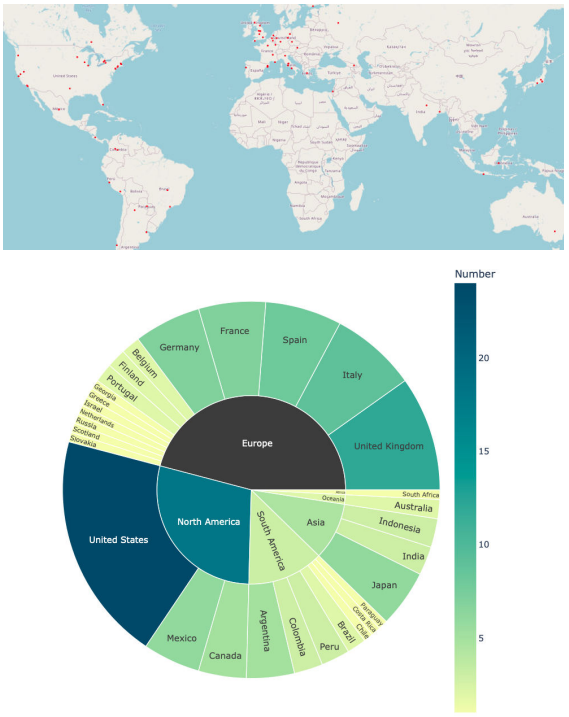


Figure 2. Approximated location of Tidal's *Longest Night* performers based on personal stream descriptions.

3. METHODOLOGY

An access to recorded archives of the event was provided by Bernard Gray, and has since been archived on Archive.org as well as Youtube [9]. Due to the uncertainties of internet networks, and some failures to connect and perform, nine of the total of 120 performances are missing (7.5%). The accessible footage represents approximately 50Gb of raw unedited video files. The quality of each video varies according to the broadcast and encoding format chosen by each artist. It can sometimes present audiovisual artefacts related to the transmission (missing sound, glitches). Rarely, videos have been archived even though they are, accidentally, partially truncated. Incomplete performances have been included in our analysis, as long as they communicate the gestures and techniques of a given live coder.

[6] We here use *Tidal* as a diminutive for all TidalCycles related live coding environments: TidalCycles, David Ogborn's MiniTidal, etc...

[7] `https://github.com/munshkr/muxy`: GitHub project.

[8] `https://drive.google.com/drive/folders/1JgCY6M0X4uab__HbVFG8ebvTgIMUXfRZ`: organisation documents, courtesy of Bernard Gray.

[9] See `https://archive.org/details/toplap` and `https://youtube.com/eulerroom`: Volunteer work from Scott Fradkin, assuming the role of an unofficial community archivist.

Strategies had to be found to deal with the profusion of footage to consider (112 * 20 minutes performances). A table – whose structure is detailed below – was used by Raphaël to compile generic information about each performance. It has been laid out as a quantitative analysis tool, allowing us to gather basic and general thematic information about a large number of performances (*see table*). As such, notes have been taken on various general topics as a ground for later inductive reasoning, confronting our experiences and potential biases to quantitative observations. The document will serve, for the discussion, solely as a reminder and visual help. In opting for this method, we hoped not to focus too keenly on specific performances or topics but rather to uncover general trends and patterns in the way the performative act is being approached by the Tidal community. Data from the event will be used as a basis for broader discussion.

Studying performances without the explicit consent of performers leads to ethical concerns. To address this, we have drawn only anonymised, general conclusions about our observations of the field. We only use data to the extent that it identifies a statistical/generic group of users sharing a specific approach to live coding. Live coders using their own tool will be identified only where their work has been made public through code source releases, academic papers or public communication.

	— Identification —
Priv.	Artist and description
Public	Location (country/town), lost/recorded, audio/visual, solo/collaborative
	— Code Editing —
Public	Text editor, editing paradigm (regular, Emacs, Vim, custom), editing speed, code comments, lisibility
	— Customisation —
Public	Bespoke tools, libraries and extensions. Physical controllers, external instruments and synthesizers
	— Musical Content —
Public	BPM, harmonic, melodic, rythmic and timbral features. Short musical description

Figure 3. Diagram of the observation grid used for each performance slot, thought to be respectful of anonymity.

Some biases in our analytic approach can already be acknowledged and considered. Naturally, studying performances on a surface level, focusing on a performer's interactions with live coding environments impair the possibility to provide deeper and nuanced example-based analyses, detailing how, for each performer, interaction relates to an artistic practice. We accept, for the time being, that this is not the scope of this article. Our intention, hereby, is to review and ponder on current live coding practices, focusing on the larger and *public* picture as opposed to the details. By consequence, it must be dully noted that we have consciously chosen to focus almost exclusively on observing the editing technique and manual execution of the performance, and certainly do not attempt to make value judgements on the code or sonic materials (audio samples, patches) being used or the musical outcomes.

One must also acknowledge our own involvement in the TidalCycles community, as the instigator of the TidalCycles project and its community Alex, as a performer, technical documentation editor and doctoral student for Raphaël. The writing of this article is motivated by the desire to deepen the understanding of the practice of live coding. Confronting TidalCycles as a research product [15] allows us to observe some of the complex interactions between technology and community and the emerging craft practices that emerge to bind them together.

4. PARADOX AND QUESTIONS

Historically, live coding has been presented as a scene for improvisation and experimentation in computer music performance [16] [17], proposing an alternative to skeuomorphic software designs and closed-source black-boxes (such as Digital Audio Workstations (DAWs), commercial software and digital synthesizers). This proposition was explored through a cathartic return to the simplicity and affluence of plain text, programming languages and grammars [18]. From the perspective of HCI (Human-Computer Interaction) research, live coding was used as a means to circumvent idiomatic patterns in music technology, such as the omnipresent piano-roll or timeline representation, reminiscent of early *musique concrète* or electro-acoustic studio practices that were sometimes felt as hindrances to direct musical expression on the laptop. More explicitly, live coding research has been focusing on exploring the computer – here considered as a fully-fledged musical environment – and computation itself as a valuable live musical expression framework, considering the control structures and internal logic of programs as compositional tools and suitable mental environments for the live performer [19] [4].

However, the provocative title of this article points to an apparent paradox: if live coding is an improvised practice, why do many performers not work *from scratch*? Approximately one quarter of performances during the *Longest Night* stream played from scratch, with the majority instead working on a continuum from working with a well-prepared musical structure, to adapting pre-prepared code, or to tweaking fully composed tracks. We approach this paradox to try to understand the nature and importance of improvisation in this particular art form.

Our analysis will be split in two sections: the first one exploring from-scratch practices, before considering the other largely dominant approaches to live coding performances. In doing so, we will argue that idiomatic patterns in the usage of live coding libraries or text editors are emerging which influence the way improvisation is approached by live coders.

5. ANALYSIS: NUANCES OF LIVENESS IN OBSERVED LIVE CODING PRACTICES

5.1 The Historical Promethean Live Coder

The liveness underpinning *live coding* as an art form and practice can be seen in the proclamations given by the 2004 TOPLAP "draft manifesto" for live coding. Although originally authored in a haphazard manner via largely anonymous contributions to the TOPLAP wiki [20], this document has often been used by musicians and artists as a practical guide to understand the gesture and spirit that unites the live coding scene, although more tempered and measured documents have since been published (*e.g.* the community guidelines for holding an Algorave [10]). The manifesto's significance for live coding development and legacy can easily be measured by the number of academic publications referring to it as the *de facto* introduction to the topic, despite its satirical, emphatic and at times self-contradictory style. The manifesto calls openly for the "access to the performer's mind, to the whole human instrument" and to the practice of live exploration of algorithms: "Insight into algorithms [...] No backup", stressing the necessity to lay bare the inner workings of the music being performed: "Show us your Screens [...] Code should be seen as well as heard, underlying algorithms viewed as well as their visual outcome" in order to invite musicians to expose, on the same level, their instruments, gestures (key strokes) and musical thoughts (algorithms). We should note that while this is ostensibly a *draft* manifesto, and is still hosted in a wiki form editable by anyone and therefore adaptable to changing practices, it has not been significantly edited for many years.

Contrary to the Algorave guidelines, the TOPLAP manifesto also emphasizes the "manual dexterity", "mental dexterity" and the thrilling danger associated with the act of live algorithmic programming, inviting musicians to take risks. This Promethean live coder figure, the one of the experimentalist, might be a key explanation to the global perception among researchers and artists alike of live coding as a particularly experimentation and improvisation-driven art form dedicated for computer savvy programmers/musicians. However, the reality is often more complex, and veterans or beginners alike can be brought to try coding from scratch in an event of the nature of the *Longest Night*. Knowledge of this art form's history and legacy is not of paramount importance for many, and from-scratch is not appearing as a significant marker between experienced veteran live coders and new users.

During the event, 24.8% (28 out of 113) of the slots were starting from scratch, with musicians presenting a blank slate on their text editor without any visible backup or prepared materials [11]. From-scratch has been attempted by solo and synchronised performers, both for audio and for video live coding, most often using Olivia Jack's JavaScript-based Hydra visual synthesizer [12]. Multiple environments have been used for this purpose including Tidal, SuperCollider (also hosting Tidal's sound engine, SuperDirt [13]), Sam Aaron's Sonic Pi [6] and bespoke live coding environments.

5.1.1 From Scratch? How and Why?

From scratch performances could perhaps be perceived as echoes of the original TOPLAP manifesto, with artists learning to adhere to the radical form of improvisation it described, a position embodied in the past by artists such as Click Nilson, and studied by Andrew Sorensen [21]. However, it is interesting to note that during the Longest Night, from-scratch performances were particularly prominent in performers from Latin America, continuing the strong community practice that developed largely independently in Mexico City [9]. While TOPLAP was initially founded in Europe, it was Mexico City that hosted the first regular community live coding meetings, where participants placed greater importance on from-scratch performance than their European counterparts through 9-minute performances sometimes referred to as "Mexican roulette".

Among the cohort of *Longest Night*'s performers, from-scratch live coders are the ones who correspond the most to the stereotypical figure of the *Promethean live coder*, but are also, by many aspects, the minority. That said, the dividing line between from-scratch coding and other forms of live coding all along the event is, more often than not, difficult to draw. At most, this form of performance can be described as a performative intention, forcing the musician to type actively. Editing speed, familiarity with efficient code editing practices (indentation, code snippets, fast navigation), deep knowledge of the libraries or languages being used (custom functions or DSP-oriented code) were not key factors motivating musicians to engage in this demanding performing technique, for novice users seemed to try their hand at it.

For the specific case of Tidal, we argue that it is quick to learn, where beginners quickly reach musical results [22]. Nonetheless, from scratch performances on Tidal, even for the observed veterans, tended to be more idiomatic, centered around the software's functionality for algorithmic manipulation of rhythmic patterns. These performances are necessarily more open to particular economic patterns of use: emphasis on repetition, creative audio mangling, interference between combined sequences, as well as fast tempos and minimal usage of parametric envelopes or complex audio effects. This observation is analogous to ones made by Thor Magnusson, presenting DMIs as "epistemic tools" embodying their own musical theory and their own usage patterns [23].

5.1.2 Celebration and collaboration

From-scratch collaborative live coding is a practice that has always been present in the community [24], and was explored during 7 different performance slots for networked collaborative jams. These slots were occupied by performers willing to celebrate the Longest Night by coding among

[10] `https://github.com/Algorave/guidelines/blob/master/README_en.md`: Algorave collaborative guidelines.

[11] We took into account performers willing to start from an initial barebone skeleton as long as it stays basic in design.

[12] `https://github.com/ojack/hydra`: GitHub repository.

[13] `https://github.com/musikinformatik/SuperDirt`: Tidal audio engine, GitHub repository.

Figure 4. A new created live coding session on David Ogborn's Estuary platform, with six performer slots.

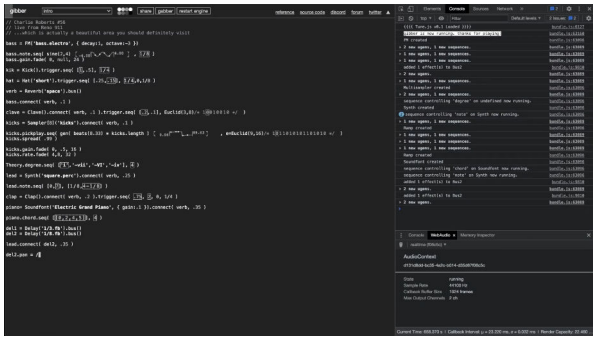


Figure 5. Charlie Robert's performance using Gibber, a JavaScript browser-based audio-visual live coding environment from-scratch during Tidal's *Longest Night*. [7]

others using David Ogborn's MiniTidal and Estuary platform [14], or by connecting privately using Flok (TypeScript and NodeJS peer-to-peer collaborative editor) or the Python-based Troop [13].

In this context, we observed that the emphasis on fast typing speed and occasional angst associated with solo from-scratch performance tend to fade. The prevalence of the format might find sources in the fact that it exposes musicians to a particularly joyful form of collaborative improvisation typically found in many other improvised musical cultures and circles, here redeployed for code-based performance. Single performers alone can already – if they do not refrain from it – have a dramatic impact on the music being generated (long samples, high gain, rhythmic density, etc...). This leads to musical behaviours already well known such as call and response, different 'band-members' focusing on different timbres/voices, or collaboration towards building complex sonic or visual algorithms. From-scratch collaborative coding exposes the musician to a group dynamic, where personal artistic intent or even self-identity as a performer is temporarily dissolved, encouraging experimentation, radical improvisation and a touch of letting go. The possibility of sharing the program state and tempo with other musicians appear to slowly lead to the establishment of this form of live coding performance as a separate sub-genre, typically showcased in networked events like the *Longest Night*. However, as is the case with environments not previously bent towards specific musical practices and styles, a certain factor of homogenisation is notable, given the limited pool of available audio samples, synthesizers and local networking capabilities given by the containerised web application.

5.1.3 Solo From Scratch

Twenty one of the from-scratch slots were occupied by solo performers, an approach to live coding that is perhaps most commented on in the literature despite being in the minority of performances during events like the Longest Night. Challenges associated with this practice are pertaining to the creation, in limited time/energy, of a complete performance complex enough to hold the audience's attention. It is also the one that more deliberately exposes the musician to errors, bugs and to the "decoupling of embodiment" [25] associated with writing code for further evaluation. On the other hand, it is also appreciated as a form of expression that offers chance events and surprises to occur, with improvisers inviting the audience to enter into an impromptu exploration of system affordances. It is not surprising to acknowledge that the technique has been used, during the *Longest Night*, to produce a wide range of distinct sonic results: ambient sample-based landscapes, breakcore/IDM music, free-form improvisation, noise, live DSP or external synthesizer-based performances. Adding to the variety of music, this form of from scratch allows live coders to showcase custom environments, or detach themselves from the most idiosyncratic type of usages, as seen through Alex performance using Vortex (Python-based Tidal rewriting) or Charlie Roberts using Gibber (Fig. 5).

As previously said, the diversity of profiles starting from-scratch exceeds our initial expectations. These performers share few common characteristics, whether it is by the environment or the interface used (from 'vanilla' to heavily modified or custom), by the nature of the code itself (musical patterns and/or lower-level DSP generative code) or by the degree and speed of interaction with code (from hardly any interaction at all to extreme velocity of edits). It is true that from-scratch coding appear to be of interest for live coders deeply versed in live digital signal programming (e.g. with JITlib [26]), hacking of custom live coding systems [27] and control of digital or hardware setups. It is also true to note that a greater familiarity with coding, programming languages, complex sound synthesis software (such as SuperCollider and its JITLib library, perhaps the most common system used for synthesis in live-coding) and text editors can be found among performers attempting this type of performance. Nonetheless, they only represent a small percentage of the performances being displayed during the *Longest Night*.

5.1.4 From scratch: some conclusions

Our observations reveal the participation of a surprisingly broad range of from scratch practices. The dangers and technical issues naturally arising from such performances, incarnated by the mythical figure of the Promethean live coder, did not act as a hindrance but, on the contrary, may have encouraged a number of musicians to embrace live

coding as a particularly radical form of computer-based musical experimentation. We were particularly surprised by the various technical profiles of musicians who lend themselves to the exercise, the practice not being limited to the most experienced live coders or computer programmers.

This phenomenon might be explained by the prominence given by live coding language designers and by live coding research in general to the development of terse and rich DSLs [14] for music expression, allowing temporal semantics, generative sequencing capabilities and precise parameter control to be expressed in a terse and legible manner. From-scratch is often made possible by the close coupling between live coding libraries and bespoke audio backends, alleviating the need for complex preparation and technical setup [15] , we noticed a stronger tendency towards musical homogenisation regarding performances produced using the from scratch non-prepared environments, linked to idiomatic patterns of software usage and livecoders relying on reflexes and procedural memory [28].

5.2 Distributing Liveness

5.2.1 Algorithmic gardeners

The rest of the performances (*approx.* 75%, 85 out of 113) featured prearranged setups and musical/video materials. The nature of the live interaction in these performances differed greatly compared to the from-scratch performers, whether they were coding collectively or solo, audio or audio/visual performances. However, it does not follow that these artists were not live coders. Rather, we could argue that this majority of the cohort prefer to distribute liveness to different stages of their work, that is, to disseminate live coded interactions more privately, across composition and preparation as well as music performance, one example being the showcase of *Wags*, a new Tidal-inspired but not Haskell-based live coding environment, by Mike Solomon [29]. This suggests a nuanced and difficult to pinpoint spectrum of live coding practices, ranging from almost totally fixed performance to paving a way, beforehand, to a more personal improvisational process, in a safer and perhaps less threatening private space.

The potential of a compositional and / or preparational phase to live coding connects it with the world of more classical generative and procedural music, where code is written and then left to produce music autonomously. In conversation with David Toop, Brian Eno explains that "generative music is like trying to create a seed". He relates generative music to gardening, "it responds to conditions during its growth and it changes and it's different every year." [30]. The assumption here is that the algorithms generating music are akin to nature, but are separate from us. Concerning live coding, one should be cautious with the generative-gardening analogy, which could be seen as separating the author from their work, where the former is cast as the genius designer of the latter.

In Eno's notion of generative music, there is distance between musicians and their algorithms. However, the gardener is also alive in the garden, working in the weather, never sitting back to watch plants grow. In *Farming as performance*, Dominic Glover builds a view based on the observation that a crop is not the outcome of a plan [31]. The farmer does not simply plant a seed and sit back. Rather, a crop is the result of an improvised performance, with both plants and farmers as performers, where any plan evolves over time. He follows the observation of Paul Richards [32] that the eventual layout of a field is not a design, but rather a partial historical record of continual responses that the farmer and their crops make to changing weather. When a live coder brings pre-written code a performance, we could see that code as a similar, living historical record.

Live coding performers, even for the case where prepared materials or setups are presented, challenge the assumptions of generative music by working hands-on with code, while it runs. At any point, the state of the code gives a snapshot of the music being generated at the present moment, allowing a performer to react through code comments, and a few keypresses might result in dramatic shifts in musical structure. By performing with code in this way, the live coder is compelled into humility, because just as with the weather, or plant life, code operates in ways and at scales beyond human control. The human performer develops heuristics for working with their code, but in a very real sense, the code also acts upon them. Prepared performances from Tidal's *Longest Night* showcase a type of improvisation much akin to what, in other terms, is described by Palle Dahlstedt as "systemic improvisation" [33].

Glover's analogy of farming-as-performance throws light on comparisons of from-scratch and prepared performances. Whereas from-scratch performers like to plant and interfere with the genetics of a seed while it grows, pre-prepared live coders instead like to bring plants to the party, and see how they fare in the performance environment, here akin to the farmer responding to weather. The key difference between algorithmic/generative music and distributing liveness in live coding performances is therefore to be found, in our opinion, in what the Longest Night allows us to observe: the observable presence of source code as a communicative medium, as a social link between the musician and their community, and as a testimony of one's implication in the unfolding algorithmic process.

5.2.2 Handling obstacles to live algorithmic performance

There are also pragmatic potential reasons for the *Longest Night* musicians tendency to pre-prepare musical materials, which go beyond and sometimes influence, afterhand, the aesthetic discourse on the practice. We observed that most of the performers belonging to this category tend to consider algorithmic code as a malleable interface, limiting their interventions to actions such as commenting lines, editing variables, crafting transitions between patterns or making terse edits to previously written materials. Reasons for doing so could be related to relative musical inexperience; limited by text editing speed, by the fear of running into bugs and performance-stopping issues, or to being ex-

[14] *Domain Specific Language*: specialized computer language for a particular application domain (*i.e.* algorithmic music, live expression).

[15] In particular, one could argue that the live coding library itself acts as a non-neutral *prepared code* for the live-coder to perform with.

posed to complexity not manageable in realtime.

On the other hand, an experienced musician may simply be unwilling to give up the levels of control they are used to in more conventional software (DAWs, mixing/mastering tools), wanting to spend significant time mastering samples and adjusting the mix of their otherwise live coded compositions before performing them. For most, our external observations are insufficient to assert the true intention behind the artists' actions, and further investigation is needed, through field interviews.

One other area for future investigation is the amount of customization which artists apply to different parts of the default Tidal system of text editor, pattern language and sound synthesizer and samples. Some take time to get deep into the default configuration of the Atom programmer's editor, working with more advanced features such as multiple cursors when coordinating a change across multiple patterns at once. Others explore alternative approaches to text editing [34], picking more obscure editors, or experimenting with live visualisation. The influence of the personalisation of the work environment on music playing for the case of the live coding scene has, until now, been too rarely addressed.

Due to the open and welcoming nature of the Longest Night event, many participants will be doing their first ever live coding performance, or even first ever performance of any kind, and still managing to find new juxtapositions of sounds and patterns transformations in the default install and sample set. There could be some concerns that the large number of performances which appeared visually similar, where artists have chosen to keep similar editor themes, could run counter to live coding's promoted ideals for freedom from constraints, that we saw in the TOPLAP manifesto. Nonetheless, the combinational possibilities offered by a live coding environment like Tidal, which offers a large number of pattern transformations which can be quickly combined to find new possibilities, means that a singular shared performing environment does not always result in similar musical results.

Pre-arranged performances from the Longest Night can prove particularly disconcerting for the analyst, as they reveal a whole spectrum of performing practices ranging from private studio work presentation to loosely arranged improvisation sets. The nature of this performing art form pushing musicians to work, perform and share with the public using mostly plain-text, improvisational and studio practices often appear blended in plain-sight, something that is, to our knowledge, not often the case for computer-based audiovisual performances.

6. CONCLUSIONS

As can be seen from our observations from the *Longest Night*, the divide between from-scratch performances and pre-arranged performances only acts as a surface typology, usable for work purposes only. From an analytic standpoint, its validity holds to the mere fact that it allows us to uncover, upon closer examination, finer distinctions between different user groups sharing the same approach to source code and live performance.

While a certain taste for liveness, demonstration and algorithmic unfoldings unites the field of live coding performance, an event of this sort appear to be a good tracker to track the evolution and diversity of live coding practices, starting from the idealized 2004 promethean live coder figure to the contemporanean and much different networked live-coder here observed during the event. Further investigation centred around the question of liveness (*e.g.* Auslander [35], Emmerson [36], Croft) will undoubtedly help us to include live coding performances in the larger field of electronic music performance.

Acknowledgments

I (Raphaël) would like to thank Laurent Pottier (ECLLA, Université de Saint Étienne) and Alain Bonardi (MUSIDANSE, IRCAM, Université Paris 8) for their advices and for their trust in this research project. I would also like to thank the 3LA doctoral school (ED 484) from the University of Lyon and my laboratory, the ECLLA from the Université of Saint Étienne, for funding this research. I would also like to thank Alex McLean for diving with me in the complex issue of analysing live coding practices.

I would like to thank Raphaël for inviting me to contribute to this paper and for his tireless support of the TidalCycles community. My work here was supported by a UKRI Future Leaders Fellowship [grant number MR/V025 260/1].

7. REFERENCES

[1] A. McLean, "Making programming languages to dance to: live coding with tidal," in *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design*, 2014, pp. 63–70.

[2] P. Rein, S. Ramson, J. Lincke, R. Hirschfeld, and T. Pape, "Exploratory and live, programming and coding," *The Art, Science, and Engineering of Programming*, vol. 3, no. 1, pp. 1–1, 2018.

[3] S. Wilson, D. Cottle, and N. Collins, *The SuperCollider Book*. The MIT Press, 2011.

[4] G. Wang, P. R. Cook, and S. Salazar, "Chuck: A strongly timed computer music language," *Computer Music Journal*, vol. 39, no. 4, pp. 10–29, 2015.

[5] A. C. Sorensen, "Extempore: The design, implementation and application of a cyber-physical programming language," Ph.D. dissertation, College of Engineering and Computer Science, The Australian National University, 2018.

[6] S. Aaron and A. F. Blackwell, "From sonic pi to overtone: creative musical experiences with domain-specific and functional languages," in *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design*, 2013, pp. 35–46.

[7] C. Roberts and J. Kuchera-Morin, "Gibber: Live coding audio in the browser," in *ICMC*, vol. 11, 2012, p. 6.

[8] A. Ward, J. Rohrhuber, F. Olofsson, A. Mclean, D. Griffiths, N. Collins, and A. Alexander, "Live algorithm programming and a temporary organisation for its promotion," in *readme - Software Art and Cultures*, Jan 2004, pp. 243 – 261.

[9] H. Villaseñor-Ramírez and I. Paz, "Live Coding From Scratch:The Cases of Practice in Mexico City and Barcelona," in *Proceedings of the 2020 International Conference on Live Coding (ICLC2020)*. Limerick, Ireland: University of Limerick, Feb. 2020, pp. 59–68.

[10] N. Collins and A. McLean, "Algorave: Live performance of algorithmic electronic dance music," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2014, pp. 355–358.

[11] A. McLean and R. T. Dean, *The Oxford handbook of algorithmic music*. Oxford University Press, 2018.

[12] C. Alexandra, "What is algorave?" https://media.ccc.de/v/rc3-2021-chaoszone-345-what-is-algorave, Dec 2021.

[13] R. Kirkbride, "Troop: a collaborative tool for live coding," in *Proceedings of the 14th Sound and Music Computing Conference*, 2017, pp. 104–9.

[14] D. Ogborn, J. Beverley, L. N. del Angel, E. Tsabary, and A. McLean, "Estuary: Browser-based collaborative projectional live coding of musical patterns," in *International Conference on Live Coding (ICLC)*, vol. 2017, 2017.

[15] A. McLean, "Research products," Mar 2021. [Online]. Available: https://slab.org/research-products/

[16] K. Burland and A. McLean, "Understanding live coding events," *International Journal of Performance Arts and Digital Media*, vol. 12, no. 2, pp. 139–151, 2016.

[17] N. Collins, "Live coding of consequence," *Leonardo*, vol. 44, no. 3, pp. 207–211, 2011.

[18] D. Stowell and A. McLean, *Live Music-Making: A Rich Open Task Requires a Rich Open Interface*. London: Springer London, 2013, pp. 139–152.

[19] T. Magnusson, "Algorithms as Scores: Coding Live Music," *Leonardo Music Journal*, vol. 21, pp. 19–23, 12 2011.

[20] C. Haworth, "Technology, Creativity and the Social in Algorithmic Music," in *The Oxford Handbook of Algorithmic Music*, ser. Oxford Handbooks. Oxford University Press, pp. 557–582.

[21] A. R. Brown and A. Sorensen, "Interacting with generative music through live coding," *Contemporary Music Review*, vol. 28, no. 1, pp. 17–29, 2009.

[22] A. McLean and R. Bell, "Pattern, code and algorithmic drumming circles," in *Proceedings of the Fourth International Conference on Live Coding*. Madrid, Spain: Medialab Prado / Madrid Destino, Jan. 2019, p. 353. [Online]. Available: https://doi.org/10.5281/zenodo.3946174

[23] T. Magnusson, "Epistemic tools: the phenomenology of digital musical instruments," Ph.D. dissertation, University of Sussex, May 2009. [Online]. Available: http://sro.sussex.ac.uk/id/eprint/83540/

[24] J. Rohrhuber, A. de Campo, R. Wieser, J.-K. van Kampen, E. Ho, and H. Hölzl, "Purloined Letters and Distributed Persons," in *Music in the Global Village Conference 2007*.

[25] S. Knotts, "Live coding and failure," *The Aesthetics of Imperfection in Music and the Arts: Spontaneity, Flaws and the Unfinished*, p. 189, 2020.

[26] J. Rohrhuber and A. de Campo, "Just in time programming," *The SuperCollider Book. MIT Press, Cambridge, Massachusetts*, 2011.

[27] B. Bacot and C. Canonne, "Musique et hacking : de l'éthique aux pratiques."

[28] T. Sayer, "Cognition and improvisation: some implications for live coding," in *First International Conference on Live Coding (ICLC2015)*, 2015.

[29] M. Solomon, "Functional reactive programming and the web audio api," in *Proceedings of the International Web Audio Conference*, ser. WAC '21, L. Joglar-Ongay and S. et al., Eds. Barcelona, Spain: UPF, July 2021.

[30] D. Toop, *Haunted Weather: Music, Silence, and Memory*, main edition ed. Serpent's Tail, 2005.

[31] D. Glover, "Farming as a performance: A conceptual and methodological contribution to the ecology of practices," vol. 25, no. 1. [Online]. Available: http://journals.librarypublishing.arizona.edu/jpe/article/id/2066/

[32] P. Richards, "Agriculture as Performance," in *Farmer First: Farmer Innovation and Agricultural Research*, R. Chambers, A. Pacey, and L. A. Thrupp, Eds. Intermediate Technology Publications, pp. 39–42.

[33] P. Dahlstedt, *Action and Perception: Embodying algorithms and the extended mind*. United Kingdom: Oxford University Press, 2018, pp. 41–66.

[34] J. Armitage, A. McPherson *et al.*, "The stenophone: live coding on a chorded keyboard with continuous control," in *International Conference on Live Coding*, Morelia, Mexico, 2017.

[35] P. Auslander, *Liveness: performance in a mediatized culture*, 2nd ed. London ; New York: Routledge, 2008, oCLC: ocn144770575.

[36] D. Peters, G. Eckel, and A. Dorschel, Eds., *Bodily expression in electronic music: perspectives on reclaiming performativity*, 1st ed., ser. Routledge research in music. New York: Routledge, 2012, no. 2, oCLC: ocn666242840.

Detection of Factors Affecting Cognitive Function in Environmental Sounds

Yasuhiro Kawahara
The Open University of Japan
kawahara2@ouj.ac.jp

Hiroshi Yoshida
NTT Access Network system Laboratories, Japan

Laurent Pottier
ECLLA Laboratory, Université Jean Monnet, France

Pierre Maret
Laboratoire Hubert Curien UMR 5516, Université Jean Monnet, France

ABSTRACT

The detection of environmental sounds that affect our daily life makes it possible to assess the suitability of the surrounding sound environment to the situation. In this study, EEGs of 20 healthy adults were measured and analyzed for changes in the frequency band power and event-related potentials. Consequently, pink noise, which changes its volume periodically, was found to easily reach the primary auditory cortex and induce changes in deep brain functions. In addition, a method for detecting environmental sounds in daily life, including such acoustic elements, is proposed. This method is developed by preparing 10-second samples of each environmental sound corresponding to the categories indicated in the environmental sound dataset ESC-50, except for short-time shot sounds. Periodic volume changes were extracted using autocorrelation analysis, and the presence of pink noisiness was determined using the flatness and crest values of the acoustic property index. Consequently, air conditioning, road repair, and washing machine sound samples were selected as typical environmental sounds containing the applicable sound elements, and all of the selected samples were long-lasting continuous sounds emitted by machines.

1. INTRODUCTION

Humans achieve various purposes in their lives, including ensuring safety, by interpreting information from their surrounding environmental stimuli. By receiving and processing these environmental stimuli appropriately, humans can maintain a comfortable and efficient social life. However, in some cases, exposure to unnecessary or excessive intensity of stimuli interferes with their comfortable life.

Conventional methods of evaluating environmental noise are primarily based on psychological effects, such as "annoyance" and "discomfort," or epidemiological studies of the effects from the perspective of health damage and prevalence [1–6]. In particular, from the viewpoint of noise, little consideration has been given to the effects of low-level sounds that are not considered harmful to health or that do not cause annoyance or discomfort. In recent years, studies on the mechanism of the effects of environmental noise on living organisms, such as the relationship with cardiovascular diseases, have been reported. A study suggested the possibility of noise as a cause of acute cardiac impairment, showing that in cases of death at night, the level of noise exposure 2 h prior to death was significantly associated with cardiac-related mortality [7]. However, it is still unclear what characteristics of the sound source cause these effects on the body and mind. The sound quality was evaluated in a study using the variation coefficient of the peak value of the frequency component of respiration [8]. Another study quantified stress during noise exposure by measuring the change in salivary amylase concentration [9]. Numerous studies have examined the effects of sound on the human body, but all of them use sounds that are conventionally defined as noise, and do not consider the effects of sounds that are not loud or unpleasant. In a previous experiment, the authors found that the biological response to continuous sound stimulation changed over time. The amplitude of P50 of electroencephalogram (EEG) event-related potentials (ERPs) gradually declined with each successive presentation of a pure tone of 1000 Hz at the same interval. This suggests that the continuous input of the same tone gradually suppresses the input to the primary auditory cortex [10]. However, as the tone of the stimulus used was a sine wave, the inhibitory effect of this continuous listening might be unique to this tone. In addition, most of the studies investigating the effects of specific tones on the human body have been conducted using musical sounds, and no studies have examined the effects of sound stimuli other than musical sounds on the body using EEG as an indicator [11,12].

If environmental sounds affecting the human body in daily life can be detected, it is possible to evaluate whether the surrounding sound environment is suitable according to the situation. In recent years, there have been many reports on the evaluation of environmental sound recognition methods [13–15]. Most of the reports focus on the recognition of singularly occurring sounds; the recognition method of environmental sounds in the background has rarely been evaluated. Since the sounds that have been pointed out to affect the human body are not singular sounds but sounds with a certain length, it is necessary to consider methods for analyzing sounds with a certain duration when proposing a method for recognizing environmental sounds regarding their effects on the human body.

In this study, to consider environmental sounds that have regular changes in volume and that affect the body, the sound elements and their functioning mechanisms were

Copyright: © 2022 Yasuhiro Kawahara et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any médium, provided the original author and source are credited.

examined by examining the changes in EEG while listening to these sounds. Next, to consider the existence of environmental sounds containing the sound elements that affect human body, an acoustic analysis method was proposed to evaluate the similarity between sounds that affect human body and environmental sounds by using a set of typical environmental sound categories, and the similar environmental sound categories were shown.

2. EFFECTS OF CONTINUOUS AUDITORY STIMULATION ON THE HUMAN BODY

2.1 Methods

2.1.1 Experiment Participants

The experimental participants were 20 healthy adults (15 males and 5 females) aged 20–22 years (mean age: 21 years). The purpose of the study, maintenance of anonymity, and possibility of withdrawal from participation were explained to the participants in document and orally, and informed consent was obtained.

2.1.2 Sound Stimuli and Presentation Methods

The sound stimuli were presented for 2 min and 40 s (±5 s) for a total of 80 times. This stimulus presentation was defined as one set, and three sets were conducted using three different types of sounds (the type of stimulus sound was the same within each set). The sound types consisted of a pure tone of 1000 Hz (S_S), white noise (S_W), and pink noise (S_P). The length of each tone was 500 ms, and the inter-stimulus interval (ISI) was randomized for each pronunciation between 1.5 and 2.5 s. The sound files were in wav format (44.1 kHz, 16 bits). A presentation software (Superlab 5.0, Cedrus) was used to control the sound presentation. The timing of the sound stimulus is shown in Figure 1. The black area represents the sounding area, and the white area represents the silent area with a random span.

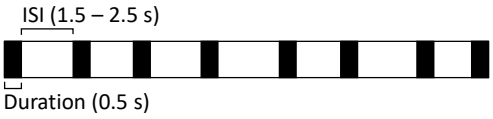


Figure 1. Timing of sound stimuli

The sound stimuli were presented by playing a file on a computer and pronouncing it via an audio interface (UA-55, Roland) with sealed earphones (hf5, ETYMOTIC RESEARCH). The volume of the sounds in the sounding parts were standardized to 60 dB SPL. This volumes was measured using an artificial ear (TYPE2015E, ACO Co.,Ltd.).

2.1.3 EEG Measurement and Analysis

EEG was measured using an electroencephalograph (Nexus10, Mindmedia). Silver-silver chloride electrodes for EEG were placed on the parietal (Cz) and occipital (O1, O2) of the head according to the 10-20 international system, and reference electrodes were placed on the earlobes. The sampling frequency was 256 Hz.

The ERPs were calculated by averaging the EEG data of the Cz site for the number of times the stimulus was presented. A duration of 1000 ms after the sound was emitted was used to calculate the average. In this study, P50 was used as an index of the intensity of the input to the primary sensory cortex. P50 was defined as the positive peak value in the 30–90 ms of the ERP waves. The values of P50 during sound exposure were compared for each stimulus condition. The power of the alpha2 band was used as an indicator of the level of deep brain activity. The alpha2 band power was calculated as the squared mean value of the EEG at O1 and O2 with a 10–13 Hz bandpass filter. For each stimulus condition, the mean alpha2 band power was calculated for 1 min before and after the stimulus presentation and during the stimulus presentation, and the values of the difference between each section were compared.

2.2 Results

2.2.1 Changes in P50 in ERP

The mean and standard error of the ERP component P50 potentials during the presentation of each sound stimulus is shown in Figure 2. The P50 potential values for each experimental participant were determined to follow a normal distribution, not significant by the Shapiro-Wilk test in any of the stimulus conditions. ANOVA showed that there was no significant difference between the means of the three groups ($\alpha = 0.05$), but multiple comparisons by the Tukey test indicate that the value when hearing SP can be considered relatively high (S_S-S_W: $p=0.85$, S_S-S_P: $p=0.24$, S_W-S_P: $p=0.08$).. In addition, Mauchly's test confirmed that sphericity of the data can be assumed.

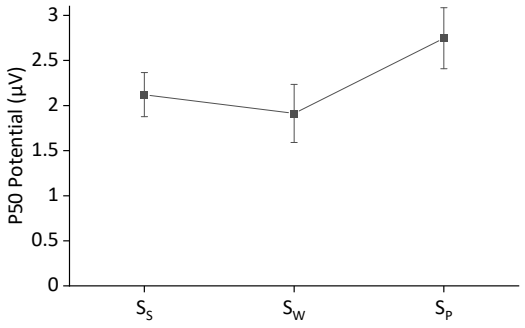


Figure 2. P50 potential during sound stimuli hearing

2.2.2 Power of alpha2 Band

The statistical values and the results of the t-test for differences of the alpha2 band power between the stimulus presentation period and the period before and after the stimulus for each sound stimulus are shown in Table 1. The values of alpha2 band power in the S_S and S_P presentation

periods were significantly lower than those in the pre- and post-periods ($\alpha = 0.05$).

	Hearing		Before/after hearing		
	Mean	SD	Mean	SD	*p*
S_S	7.84	4.67	8.35	4.50	0.041
S_W	8.23	4.85	8.27	4.73	0.876
S_P	7.47	4.41	8.21	5.17	0.015

Table 1 Comparison of alpha2 band power during hearing and before/after hearing

2.3 Discussion

The peak value of P50 reflects the input intensity of the auditory stimulus to the primary sensory cortex and is used as an index of sensory gating. Sensory gating is a filtering mechanism that works on sensory stimuli input from the external environment, which can suppress the input of unnecessary stimuli to the sensory cortex [16, 17]. In this study, the values while hearing S_P were higher than those while hearing other types of sounds. This implies that sensory gating is not easily applied to pink noise with regular volume changes and that the input intensity to the primary auditory cortex of this sound is not easily suppressed.

The EEG alpha2 powers during the presentation of the S_S and S_P stimuli were significantly reduced compared to the values before and after the stimuli. The EEG alpha2 band power is used as an indicator to reflect fundamental brain activity [18]. This result suggests that the continuous presentation of pink noise and pure tones may suppress fundamental brain activity. This may be due to fast auditory cognitive processing of stimuli by S_P, which are difficult to be sensory-gated, and therefore may suppress fundamental brain activity by making it difficult for brain functions to operate multimodally. On the other hand, S_S affected the fundamental brain functions even when sensory gating was applied, owing to the high local spectral density of energy on a single frequency compared to other stimuli.

3. DETECTION OF SOUNDS WITH PHYSIOLOGICAL EFFECTS IN ENVIRONMENTAL SOUNDS

3.1 Environmental Sounds with Repeated Volume Changes

Pink noise with regular volume changes may suppress fundamental brain function. In this section, we discuss the existence of such sounds in daily living spaces by using environmental sounds in typical living spaces. For preparing the environmental sounds to be used, 10 s of sound samples that match the category of environmental sounds lasting more than 10 s each were employed, referring to the categories of the ESC-50 dataset [19], which is widely used for environmental sound classification. The samples were obtained from the BBC Sound Effect library and from the original recordings in a living environment. The file format of the sample sounds was 44.1 Hz, 16-bit, monaural wav format, and the volume was adjusted to EBU R128 = -24.0 LUFS in all samples.

3.2 Method for Detecting Environmental Sounds that Repeatedly Change in Volume

From the 25 environmental sound samples, those with continuous repeated volume changes were extracted. The period of the repeated volume changes to be extracted was set to be more than 50 ms (less than 20 Hz), which exceeds the audible range.

To detect the repeated volume change, we correlated the self-waveform with the waveform that was shifted by various periods, and obtained the peak value as the maximum value of the correlation in 50 sampling periods (1.13 ms) before and after, and set the top five periods with high correlation as the “basic period”. Then, for each basic period detected, the correlation was calculated with a signal shifted by an integer multiple (up to 10 periods) of the basic period (e.g., 100 ms, 200 ms, 300 ms for 100 ms basic period (10 Hz)) to check whether the basic period is a regular change over the whole signal. If the change in the correlation coefficient with successive shifts from the basic period to 10 times the fundamental period is always greater than 50% of the correlation coefficient at the previous shift, then there is a regular change across the signal during that basic period. Consequently, for 5 of the 25 environmental sounds, there is a regular volume change with 12 basic periods of 2.1–15.9 Hz. An example of the correlation calculation is shown in Figure 3.

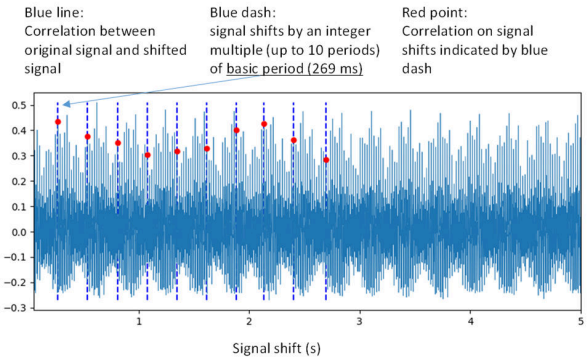


Figure 3. An example of sound regularity judgement

3.3 Method for Detecting Environmental Sounds including Pink Noise

Next, we tested whether the same environmental sound contained pink noise using the following procedure. Flatness and crest [20] were calculated as the features of each sound. In the pink noise that was repeatedly sounded at regular intervals (0.2, 0.5, and 1.0 Hz), flatness showed high values and the crest showed low values, as depicted in the values of the sound sample PinkNoise (0.2, 0.5, and 1.0 Hz) in Table 2. The flatness and crest are considered to be effective in determining the pink noise characteristics of sounds. The same report was made for a musical analysis [21].

3.4 Judgment Results of Environmental Sound Samples by the Proposed Analysis Method

Table 2 lists the fundamental frequency of the volume change for the sound sample that had a periodic volume change, among the 25 environmental sound samples. The values of flatness and crest for each sample sound are listed, with values of flatness above 0.75 and crest below 5 shown in bold. The four sound samples with both values in bold are air conditioning, rain, road repairs, and washing machine. These samples are considered to contain pink noise characteristics. Therefore, when combined with the results of the detection of sounds with regular volume changes, among the 25 environmental sound samples, 3 samples were found to exhibit regular volume changes and pink noise characteristics: air conditioning, road repairs, and washing machine.

Category	Regular volume alteration freq. (Hz)	Flatness	Crest	Like pink noise with regular volume change
air conditioning	2.08	**0.79**	**4.02**	+
airplane		0.54	**4.88**	
chainsaw		0.38	6.21	
chirping birds		0.14	45.27	
clock alarm		0.09	190.27	
clock tick		0.46	27.98	
crackling fire		**0.85**	11.11	
crickets		0.14	28.97	
crying baby		0.26	25.29	
engine		0.57	**2.93**	
farm machine		**0.80**	8.86	
frog		0.12	35.76	
hand saw		0.32	12.50	
helicopter		0.34	7.32	
insects		0.11	17.40	
keyboard typing		0.30	17.08	
rain		**0.88**	**3.99**	
refrigerator	5.36	0.59	**4.09**	
	12.07	0.59	**4.09**	
	2.30	0.59	**4.09**	
	3.71	0.59	**4.09**	
	2.84	0.59	**4.09**	
road repairs	15.90	**0.91**	**3.78**	+
sea waves		0.70	5.52	
siren		0.37	7.19	
snoring		0.55	12.77	
train		0.56	5.09	
vacuum cleaner		0.54	8.34	
washing machine	2.64	**0.78**	**4.61**	+
	2.20	**0.78**	**4.61**	+
	3.30	**0.78**	**4.61**	+
	4.40	**0.78**	**4.61**	+
PinkNoise (0.2 Hz)	-	**0.99**	**2.46**	+
PinkNoise (0.5 Hz)	-	**0.97**	**1.81**	+
PinkNoise (1.0 Hz)	-	**0.96**	**1.70**	+

Table 2. Analysis results of environmental sound samples

4. SUMMARY

By measuring EEG event-related potentials P50 and α2-band power during sound stimulation, we confirmed that pink noise with repeated volume changes is more likely to reach the primary sensory cortex and causes changes in deep brain functions than other type of noises. Furthermore, we investigated the detection of environmental sounds containing such acoustic elements to assess environmental sounds in daily life. By applying an algorithm to determine the presence of periodic volume changes and pink noise, the sound samples of air conditioning, road repairs, and washing machines were selected from the category of typical environmental sounds. All three samples were long-lasting sounds produced by the machines.

Acknowledgments

This work was supported by a JSPS KAKENHI Grant.

5. REFERENCES

[1] K. Furihata and T. Yanagisawa, “Investigation on composition of a rating scale possible common evaluate psychological effects on various kinds of noise sources,” in Acoustical Sci Tech, 1989, vol. 45, no. 8, pp. 577–582

[2] E. Öhrström, R. Rylander, and M. Björkman, “Effects of night time road traffic noise-an overview of laboratory and field studies on noise dose and subjective noise sensitivity,” in J Sound Vib, 1988, vol. 127, no. 3, pp. 441–448.

[3] L. S. Finegold, C. S. Harris, H. E. von Gierke, “Community annoyance and sleep disturbance: Updated criteria for assessing the impacts of general transportation noise on people,” in Noise Control Eng J, 1994, vol. 42, no. 1, pp. 25–30.

[4] H. M. Miedema and C. G. Oudshoorn, “Annoyance from transportation noise: relationships with exposure metrics DNL and DENL and their confidence intervals,” in Environ Health Perspect, 2001, vol. 109, no. 4, pp. 409–416.

[5] M. Basner et al., “Auditory and Non-Auditory Effects of Noise on Health,” in Lancet, 2014, vol. 383.9925, pp. 1325–1332.

[6] T. Munzel et al., “Cardiovascular Effects of Environmental Noise Exposure,” in European Heart J, 2014, vol. 35.13, pp. 829–836.

[7] A. Saucy, B. Schäffer, L. Tangermann, D. Vienneau, J.-M. Wunderli, and M. Röösli, “Does night-time aircraft noise trigger mortality? A case-crossover study on 24 886 cardiovascular deaths,” in European Heart J, 2021, vol. 42, no. 8, pp. 835–843.

[8] T. Yoshida and M. Yamada, Sound quality evaluation using the change of the peak value of the breathing,” in Reports of the meeting in Acoustical Society of Japan, 2007, pp. 765–766.

[9] A. Arimitsu, C. Wan-Ho, and T. Toi, “The understanding the state during the intellectual production activities based on physiological information,” in Proc Meeting. The Institute of Noise Control Engineering of Japan, 2011, vol. 2011, pp. 85–88.

[10] J. Ishii, Y. Kawahara, and Y. Katagiri, “Characteristics of Periodic Environmental Sounds that Affect Human Sensory Inhibition and Recovery,” in The Second International Workshop on Smart Sensing Systems (IWSSS'17), 2017 pp.12–13.

[11] J. Schlittenlacher, W. Ellermeier, and G. Avci, “Simple Reaction Time for Broadband Sounds Compared to Pure Tones,” in Attention, Perception & Psychophysics, 2017, vol. 79.2, pp. 628–636.

[12] T. R. Agus, C. Suied, S. J. Thorpe, and D. Pressnitzer, “Fast recognition of musical sounds based on timbre,” in J Acoustical Soc America, 2012, vol. 131(5), pp. 4124–4133.

[13] S. Chu, S. Narayanan, and C. -. J. Kuo, "Environmental Sound Recognition With Time-Frequency Audio Features," in IEEE Trans Audio, Speech, and Language Processing, 2009, vol. 17, no. 6, pp. 1142–1158.

[14] S. Chu, S. Narayanan and C. -. J. Kuo, "Environmental sound recognition using MP-based features," in 2008 IEEE Int Conf Acoustics, Speech and Signal Processing, 2008, pp. 1–4.

[15] Z. Mushtaq, and S.-F. Su, “Environmental sound classification using a regularized deep convolutional neural network with data augmentation,” Appl Acoustics, 2020, vol. 167(4):107389.

[16] N. Boutros, A. Belger, D. Campbell, and J. Krystal,” Comparison of four components of sensory gating in schizophrenia and normal subjects: a preliminary report,” in Psychiatry Research, 1999, vol. 88, no. 2, pp. 119–130.

[17] K. Jerger, C. Biggins, and G. Fein, “P50 suppression is not affected by attentional manipulations,” Biological Psychiatry, 1981, vol. 31, no. 4, pp. 365–77.

[18] K. Omata, T. Hanakawa, M. Morimoto, M. Honda, “Spontaneous Slow Fluctuation of EEG Alpha Rhythm Reflects Activity in Deep-Brain Structures,” A Simultaneous EEG-fMRI Study, PloS One. 2013, vol. 8, no. 6.

[19] K.J. Piczak, ESC, “Dataset for environmental sound classification,” in Proc 23rd ACM Int Conf Multimedia, Brisbane, Australia, 26-30 October 2015, pp. 1015–1018

[20] G. Peeters, et al., “The Timbre Toolbox: Extracting audio descriptors from musical signals,” in J Acoustical Soc America, 2011, vol. 130, no. 5, pp. 2902–2916.

[21] L. Pottier and N. Wang, “Analyses quantitatives de la musique par des mesures réalisées sur des signatures spectro-temporelles du son,” Journées d'Informatique Musicale, LaBRI, France, May 2019.

Sound Design Ideation: Comparing Four Sound Designers' Approaches

Rod Selfridge
KTH Royal Institute of Technology
rod2@kth.se

Sandra Pauletto
KTH Royal Institute of Technology
pauletto@kth.se

ABSTRACT

The craft of sonic interaction design (SID) for new digital objects is a non-trivial task. A deeper understanding of sound design practices employed by professional sound designers and Foley artists offers a rich source of knowledge which may inform future designs. Our research utilises a novel set of design briefs to examine and compare initial sound design ideas produced by four sound designers. Two participants are professional sound designers, while the other two are former professional sound designers who recently moved into academia. Results show a number commonality in ideas produced as well as uniqueness in approach and personal influences when completing the briefs.

1. INTRODUCTION

As new digital objects, with no inherent sound, are constantly being created the need for a deeper understanding of sound design practice and the role of the sound designer has increased. Often there are no inherent sounds associated with digital objects and multimedia product designers are often given the challenge of developing these, including bestowing some meaningful audio feedback. This is a non-trivial task and much can be learnt from practices established in media production. Sound designers and Foley artists have a history of creating sounds for everyday interactions and objects in creative industries including cinema, games, VR/AR, etc. Research into the design and production of these sounds is in its infancy, largely disconnected to its closest practice in media production. Recently, the creative contribution that sound professionals bring to multimedia production has started to be investigated (e.g., [1–3]) and results reveal the richness of sound designers' training processes, experiences and amassed knowledge.

This work contributes to this area of research by gathering new knowledge about the creative processes of professional sound designers with the view to strengthening connections between sonic interaction design and the existing practice of sound design. This research aims to inform the development of new techniques and tools for implementing sounds for new digital objects, following an approach inspired by Dourish and Button's concept of technomethodology [4] where ethnomethodology investigations form the basis of the design of new technology. New tools, informed by professional sound designers, will assist sonic interaction designers, reinforcing links between theory and what has already been developed and culturally established in creative practice. This paper presents the results of a study conducted with four sound designers, two participants off which are professional sound designers, while the other two are former professional sound designers who moved into academia. On the surface, each participant has a unique background and career path. Their ages cover different generations, there are three male and one female, each from different countries with different cultural backgrounds and work within different media (radio, film, games and sonic art). Results show, however, that common concerns, strategies and dimensions, which seem to bridge across these fundamental differences, exist along with unique approaches to identical tasks.

Copyright: © 2022 Rod Selfridge et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. BACKGROUND

Sound design is a means to convey information relating to the function and the form [5] of an object or system. The function of a sound informs the listener of an event or an interaction that has caused the sound to be created. The form of a sound informs the listener about the characteristics of the object(s) themselves, e.g., the objects' size, whether the object is old or new, etc. For example, a knock on a door tells us that someone wants to come in (function), while the quality of the sound, form, indicates whether the door is made of wood or glass, or whether the knock is sad or happy [6]. Sonic Interaction Design (SID) focuses on the role of audio and sonic feedback in interactive devices [7].

Research on how design practitioners, in sound or other areas, think and use their creativity [8] is relatively recent [4,9–12]. Schön and Dourish have emphasised the dynamic nature of these processes [4,9]. They happen "in action", and therefore require to be examined "in action" too. Empirical design research can vary between "in the studio" ethnographic studies, which have the advantage of observing practitioners in situ, but are context dependent and difficult to generalise; and laboratory studies, where tasks and protocols might be more contrived in order to control variables, but comparisons and generalisations are potentially possible. Finally, intermediary studies, which aim to maximise the benefits of both approaches, are also possible, but difficult to design and carry out. Despite the differences, all these approaches have been found to be effective in extracting new knowledge about design processes [13]. Ethno-

graphic studies [14] employ a number of techniques [15], including observations of practice and semi-structured interviews, which help researchers identify key factors and methods undertaken by individuals within their everyday processes and practices. Pauletto [2] has used these techniques to investigate the creative processes of film audio post-production professionals.

The think-aloud method [8, 16, 17], can elicit knowledge during practice, in which participants are encouraged to speak their thoughts out loud, with minimum interruption from researchers. Using think-aloud techniques, in conjunction with traditional interviews, can provide a better understanding of what participants do [18]. Previous research into SID where individuals and practices are observed have been carried out in [19]. A workshop involving postgraduate students was undertaken to analyse their sound design process, where bottlenecks occurred, and the cognitive behaviours involved in the creative process. Our research focuses on professional, rather than non-professional, sound designers currently or previously working across a range of media. We contend that the practice of professional sound designers might be richer in approach and originality than that of non-professionals, and therefore more fruitful when aiming to capture the sound designers' reflection-in-action [9] that could potentially be applied, in part or wholly, within sonic interaction design.

3. METHOD

A study procedure was developed for this project, which includes observation of practice in situ, an interview, and newly developed sound design briefs to be tackled by all our participants. A pilot study, reported in [20], confirmed that the sound design briefs we developed are successful in eliciting rich sound design creative processes.

3.1 Description of Study Procedure

A key element of our wider research is to understand how sound professionals conceive appropriate, original sound designs for a given task. To obtain research data in this area, we developed an intermediary study procedure that has both "in the studio" elements and laboratory aspects [13]. It is based on ethnographic techniques [15, 21], and think-aloud methods [16, 17, 22], which includes a set of specifically devised sound design briefs to be common across all participants so that different approaches to the same task can be compared. The overall study procedure spans a maximum of 4 days and includes the following phases:

- Day 1: Initial contact and establishment of rapport (prior to Day 2)
- Day 2: Observing moment-by-moment [4] practice (Duration: 4-8 hours)
- Day 3: Observing how sound designers approach given briefs (Duration: about 3-4 hours)
  - Brief 1: Abstract brief
  - Brief 2: Listener in focus
  - Brief 3: Case Study - The sound of air pollution
- Day 4: Final observations and follow-up conversation

The initial phase involves recruiting participants, engaging in dialogue about their role, outlining our research, and generally developing rapport. In the next phase, we observe the designers in their studio and learn about their moment-by-moment [4] responses to the daily environment and tasks, through tools and practices. The activities of Day 3 are designed to facilitate reflection-in-action [9] during the initial stages of the sound design process. Having a set of common briefs gives us an explicit way to identify common practices, as well as highlight unique approaches leading to novel solutions. The briefs also aim to challenge participants' habits, open their minds and provoke creative solutions. Sound designers are located in their normal working environment, usually a studio, and are free to use any of the tools they are familiar with to illustrate the type of sound designs they are thinking of while responding to the briefs. At the end of the briefs, researchers can seek clarifications, if necessary, without interrupting the thinking flow (a recognised technique within think-aloud methods [23]) and participants can offer feedback on the method. Finally, a semi-structured interview is conducted at the end of the process to gather more precise background information from the participant.

3.2 Description of Briefs

The briefs provide us with a tool to study sound designers, bringing to the fore their creative thoughts and processes. A previous study [20] using these briefs has shown that they are successful in eliciting a broad range of processes. This study focuses instead on the ideas produced within the briefs procedure. Non-standard tasks were deliberately chosen in order to shift the sound designers from their habits and comfort zones, pushing them to reflect on their sound design choices rather than rely on previous solutions. It also allowed us to present sound designers from different media fields (radio as opposed to film, for example) with the same brief. The Design Thinking framework [11, 24], partially influenced the way the briefs were constructed starting with a very open brief and concluding with a more focused one. Specifically, design thinking highlights the importance of exploring and empathising with the end-users and their needs, before reformulating the initial brief on the basis of this. Additionally, researchers have a list of what-if questions to be used at their discretion during the session to stimulate further design [24]. The aim of the initial brief, Abstract Brief, is twofold. Firstly, to present sound designers with an open task that challenges habits that they might have developed in their work, and secondly, it is an opportunity for participants to experience speaking out loud their thoughts in the presence of the researchers. The participant is presented with four coloured shapes and asked to choose one. This is the character they need to create sound for, which is described as walking happily down the road. On completion

of this, participants are informed that the character has arrived at the end of the road and that they are waiting for their friends, who are however not coming, so the character is now sad.

The second brief, Listener in Focus, is designed to encourage the designer to explicitly consider the listener of the sounds being created. Designers are asked to consider one specific group of listeners, and then, once finished, on a different group of listeners, who may (or may not) be judged to have different values from the first group. Participants are asked to consider how they would create the sound of a thinking, active brain that will be listened to by a group of engineers in order for them to understand it. On completion of this, designers are informed that the brain is now to be listened to by a group of people who are vegans. Obviously, engineers and people who are vegans are not mutually exclusive groups and indeed, sound designers can decide that their sound is appropriate for both groups. However, they could also decide to adapt their design for an audience depending on what characteristic of that audience has been highlighted to them. Brief three, Case study - The sound of air pollution, is based around a specific, more concrete, scenario of a coat with an embedded air pollution sensor. This brief purposely presents media production sound designers with what could be defined as a sonic interaction design task. In this brief, the air pollution sensor produces a sonic output which informs the wearer about the quality of air at their location. A route map of a journey in a city is given, and participants are invited to sketch the coat's sounds. On this occasion, researchers aim to observe how participants tackle a brief that is presented more concretely.

In the pilot study [20] thematic analysis was carried out where we identified a total of 8 themes and 3 sub-themes that successfully portray the main articulations of the creative process. Initially, themes were derived from the data itself rather [25]. We then compared our themes with those, for general design processes, devised by [26] adapted from [27] and validated in [13]. Our final themes adopt some themes from [26] as well as add some themes that emerged from the pilot and can be seen in [20]. The pilot study highlighted the processes, whereas in this paper we examine the ideas themselves, comparing and contrasting the four sound designers approaches.

3.3 Data Analysis

Audio recordings were transcribed. Otter.ai provided a first rough transcription, and then we refined this transcription by re-listening to the audio. The transcriptions were coded using NVivo using the concepts of Shaping Ideas and Incremental Ideas from [28]. Shaping ideas are defined as appearing usually early in the process and as being influential on subsequent ideation, while Incremental Ideas are defined as being single ideas that make small additions connecting one idea to the next, and are closely tied to the previous ideas. This allowed us to compare sound designers in terms of idea generation. After this, by analysing the sound design ideas' content, we highlighted the main directions along which sound designers seem to develop their initial sound design solutions. We organised and named overall 15 dimensions which are described in Section 4.

3.4 Participants

3.4.1 The Professionals

Participant 1 (hereafter P1) has over 35 years as a sound engineer working within radio drama for Swedish radio. When he took up this role, there were no educational programmes in the subject area and all knowledge had to be learned on the job. Although he has a wide experience in all aspects of sound (sound design, recording, editing, mixing, etc.), it is only until relatively recently (3 – 4 years) that he has taken over responsibility for performing Foley. Participant 2 (P2) is a sound designer who works mainly in independent, artistic and experimental films (feature and shorts), but she also develops sound design for other media such as VR, and theatre. She has been a successful freelance sound designer since graduating from a Master in sound design in 2012 in the UK and carries out the roles of sound designer, sound editor, mixer and Foley artist as required.

3.4.2 Former Professionals

Participant 3 (P3) has a professional background in sound for games where he had the role of Sound Designer and Sound Engineer for a start-up company. He is now a doctoral researcher whose interests include sound in interaction, interaction design, and human-robot interaction. Participant 4 (P4) is an artist whose work focuses on sound, music, physical interaction, games, and building new instruments. He has worked as a professional sound designer on an array of independent projects. He is now a doctoral researcher focusing on supporting sustainability through sonic interaction design.

4. RESULTS

The coding of Shaping and Incremental ideas for the four sound designers shows some large differences between participants. P2 produced about 3 times more ideas than P1, for example (see Fig: 1). The ratio between number of Shaping ideas and number of Incremental ideas is about 1 to 2 for all sound designers. P1 and P4 have produced a similar amount of ideas across briefs. P3 has produced more ideas for Brief 2 than the other two, P2 has produced the least number of ideas for Brief 2 than the other two.

An analysis by the two authors of the sound design ideas' content highlighted 15 *Dimensions* along which sound designers articulated their thoughts (see Table 1). These are grouped into 3 categories: Overarching strategies sound designers seem to adopt irrespective of the nature of the brief; Design Strategies adopted when tackling a specific brief; Approaches to the creation of sound. Here follows some examples from the transcripts of the articulation of each dimension.

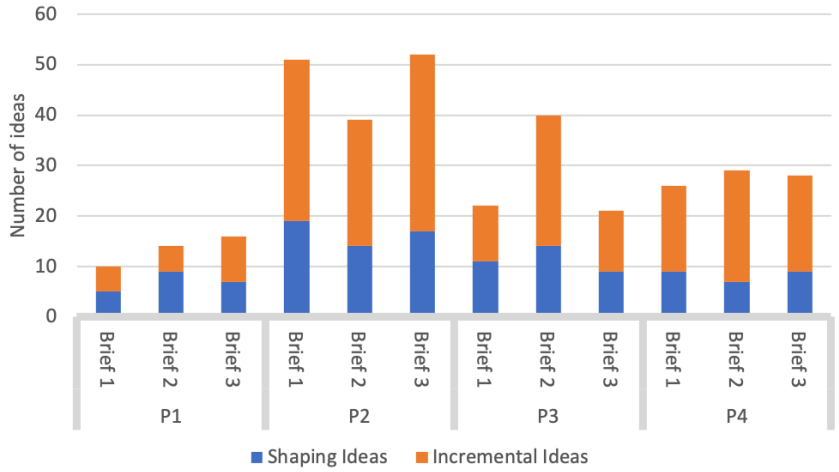


Figure 1. Graph illustrating the number of different ideas from the participants, separated into shaping and incremental ideas [28].

Table 1. Emerging Dimensions

Dimension	
Overarching Strategies	
Audiovisual Construct	Visualisation vs. Audio Only
Knowledge	Research vs. Culture
Anthropomorphism	Human vs. Non-human
Future	Speculation vs. Not Considered
Guidance	Directed vs. Autonomy
Design Strategies	
Initial Focus	Scene vs. Object
Mapping	Literal vs. Association
Characteristics	Static vs. Dynamic
Adaptation	Objective vs. Subjective
Design Approach	The Sound vs. The System
Context	Dependent vs. Independent
User Control	Provided vs. Absent
Approach to Sound	
Sound Provenance	Synthetic vs. Real
Audio Effects	Processed vs. Unprocessed
Communication	Existing vs. Novel
Perceptual Issues	Considered vs. Disregarded

4.1 Overarching Strategies

4.1.1 Audiovisual Construct - Visualisation vs. Audio Only

This refers to whether sound designers tend to visualise what they are trying to design sound for, or whether they tend to remain in an audio domain while ideating. Throughout the sound design briefs, P2 seems to conceptualise sound only as audio. For example, she does not speculate of potential visual characteristics of the object producing the sound, or picture a scene or context in which she sound might be occurring. One example of this is the sounds of the shape happily walking down the road. P2 does not discuss the scene or context at all, instead ideates purely about the sound of the shape and designing audio for it. Similarly, P1 describes a scene as in terms of a soundscape rather than anything visual. On the other hand, P3 describes the brief as an audiovisual scene, and P4 examines visually the context in which the shape is meant to sound, *"... how are we seeing this happen? ... is it like a two-dimensional road?"*.

4.1.2 Knowledge - Research vs. Culture

This refers to the tendency to research the objective characteristics of an object (its physical characteristics, how it works, etc.) in order to inform the sound design, versus getting inspired by what the object might be culturally associated with. When designing the sound for a brain belonging to a snake, P2 states that she would carry out research to gain an understanding of the functions of a snakes' brain. She envisaged the sounds created would be designed from this knowledge, hypothesising that the functions of a snakes' brain will be different in size and scale from those of a human brain. This is in contrast to P3's approach, which tends to make use of sounds already culturally associated to a snake to create a new sound design. His comments include, *"there will be lots of..., more like hissing sound, of a snake, because that's definitely associated with snakes"* and *"Rattle snakes, they have, the tail have this kind of rattling sounds. That, that could be there ..."*. Another example of this is can be found when P1 is describing the worst sound design that could be employed for the sound of a brain. He states, *"boiling pottage (porridge) ... but maybe that's not totally wrong, I don't know."*. Porridge Brain Syndrome is a anecdotal condition where the person suffering complains of excessive forgetfulness, tantrums and poor memory. It is often described on pregnancy forums [1] .

4.1.3 Anthropomorphism - Human vs. Non-Human

This is the tendency for some sound designers to give human characteristics such as emotions and behaviours, to

[1] https://4mom4ever.blogspot.com/2015/10/porridge-brain-syndrome.html

the object for which they are asked to design a sound for even when the object is not human. In our study, this strategy appears on a number of occasions, more for some sound designers than others. This is evident in P3's approach to the first brief where he imagines a red square as a woman, red being the colour of her dress and square being her movements. This is also seen in P1's approach to the same task, where he adds footsteps and foot shuffling to express the shape's movement and feeling. In contrast, when ideating about sounds of the shape in brief 1, P4 keeps the character as a shape but he gives it some humanoid characteristics when the shape is meant to be unhappy, by vocalising a sigh-type sound accompanied by hunching shoulders (in an imagined animation). Finally, P2 does not humanise any of the sounds she designs throughout the briefs.

4.1.4 Guidance - Directed vs. Autonomy

This refers to whether a sound designer searches for additional guidance from others in the project (director, producer) or from other documents (script), or they are comfortable to define all aspects of the project themselves if given the freedom to do that. In our study, the sound designers varied in how much autonomy in sound ideation they felt comfortable with, and when they would seek further direction from other sources. P1, P3 and P4 all looked, at least initially, for additional information regarding the context and the object of brief 1. P1 indicated, after a few sentences, that he would be looking for a script to gauge the pace, the mood and how the object is supposed to be moving in order to develop his idea of footsteps any further. P3 initially thought of the sounds of footsteps, along with birds, people chattering and music before stopping. After eliciting details on the footsteps he stated, *"if I only got that sentence to begin with, I would stop there because usually I would ask whoever person designing the scenes for more details"*. P4 looked for clarification early on while ideating about his sound design for brief 1 stating, *"Am I in charge of the animation as well? Or can I just be in charge of the sound? Okay, I'm in charge of everything. Yes, I'm a vengeful and terrible God."*. P2 did not seek the same guidance and focused straight away on realising the sound she had initially described.

4.1.5 Future - Speculation vs. Not Considered

This refers to the tendency to speculate about what the future would look like and base ideas for sound design on that speculation, when a brief is set in the future. During brief 3, the sound designers were asked to ideate a sound for a coat that can detect air pollution if the year was 2200. Both P1 and P2 considered that the coat would convey extra information to the user. Similarly to P1, P2 includes in the design a voice like Siri as well as beeps or Morse code. P2 then moves on to the idea of a musical output. Overall, P1 and P2 do not speculate greatly about the future. On the other hand, P4 reflected that the harmonic and rhythmic preferences of the future might be different. Since his design relied on environmental sounds as input, the overall sonic result would be different because including futuristic sounds from the environment.

4.2 Design Strategies

4.2.1 Initial Focus - Scene vs. Object

This refers to the tendency, when approaching a brief, to imagine and describe an overall scene, rather than concentrating on the sounds of the object within the scene. Typically, P1 and P3 would first ideate a scene and its soundscape thinking about footsteps, traffic, etc. Additionally, both indicated that they would be looking for further information regarding the scene. P1 states: *"it's very depending on the dialogue"*. P4, on the other hand, initially reflects on different potential scenes, *"Am I like a first-person hexagon walking through, walking down a road? Or is it like a third person hexagon? Or, like, how far would the camera.... is this on a screen?"*. After establishing the scene, his focus is on the sounds for the objects rather than the scene. For all the briefs, P2 focuses on the object from the outset, stating at the beginning of brief 1, *"I'm thinking about the shape in isolation, so not thinking about the task, per se, like 'The walking down the road' "*.

4.2.2 Mapping - Literal vs. Association

There are clearly different approaches when ideating about sounds relating to specific objects and the paths that our participants chose to explore. Some looked for literal interpretations of the objects while others looked for an association to the object. An example of this can be found when participants were asked to design the sound of the brain of a superhero. P1 thought that the sound of a superhero's brain should have an association with the specific superhero's powers, or their signature theme tune. P3 has very similar ideas, using the character's voices (*...in a low deep voice saying, "I'm Batman"*), or including the superhero's powers into the sound design, e.g. Superman hearing all the sounds throughout a city. In contrast, P2's approach is to start from a data set of brain waves and shift the frequency content to higher frequencies so that it is audible, and then introduce additional frequencies that brains of ordinary people don't have to signify that the brain is that of a superhero. Different approaches can be found also for brief 3. For P1, more pollution required an alarm sound, so he considers different kinds of beeping sounds, or a Geiger counter sound: the rate of beeping would increase as the air pollution increases. This was then complemented by the use of a synthesised voice like *"Siri"* that informs the coat wearer of the air pollution level. P3 associates poor air quality to the sound of breathing while blocked up with a cold. In contrast to this, P2 associates poor air quality with a more noisy sonic output, describing a sound design based on filtered noise where the narrower the filters the cleaner the air. P4 had a similar approach where he associated poor air quality with a *"dirty (sonic) aesthetic"*.

4.2.3 Characteristics - Static vs. Dynamic

This refers to focusing more on the dynamic characteristics of a brief, rather than the static characteristics. In brief 1, P2 and P4 developed the sound of the shape on the basis of how they imagined the object rolling down a hill, or imag-

ining how they would move over obstacles or how they would bow their head when sad. In contrast, P3 designed the sound on the basis of associations he made with the objects static characteristics. He chose a square, and then associates its lines to straight and assertive movements.

4.2.4 Adaptation - Objective vs. Subjective

This refers to the tendency to consider a sound design, when finished, as something objective, that cannot be altered on the basis of a different listener, or a different context, or a different highlighted quality of the object producing the sound. An example of this can be found in the ideations of brief 2. P4 described a sound design for a brain to be listened by engineers. When asked what the worst sound for a brain would be, he stated that he would make the initial sound design "less accurate". He did not attempt to vary the main concept of the sound design he ideated, but considered what the word "worst" meant within that concept. On the other hand, P3 did not think that once created a sound design could not completely diverge from the original concept. When asked the same question, to ideate the worst sound for a brain, he completely changed to a new type of sound based on words associated with brains (*"Brain-fart"* and *"Brain-freeze"*), and continued by describing how to make the sound of freezing.

4.2.5 Design Approach - The Sound vs. The System

This refers to the tendency to approach a sound design by ideating a sound producing system first, rather than directly ideating the sound. On a number of occasions, a sound designer would describe a system which would generate sounds based on mappings from input to output, rather than a sound to represent the object in the brief. An example of this is appears when the designers were invited to describe the sound of a brain to be listened to by vegans. P4 discusses a system for generating the sounds of a brain based on amplifying and shifting the waves of the brain. Therefore, when the additional *what-if* question was asked to describe the sound of the brain on drugs, for example, P4 replied *"I don't know ... I wouldn't want to know ahead of time"*, meaning that he would input the waves of a brain on drugs and accept whatever result the system would produce. This is in contrast to the other sound designers who designed the sounds for a brain and then adapt the sounds based on their impression on how the brain would sound under the influence of drugs (e.g. the sounds would become more confused, erratic, etc.).

4.2.6 Context - Dependent vs. Independent

Some of our participants strongly linked the ideated sound designs with the context they imagined for the brief. For example, P4 created a sound for a coat that communicated air pollution using sounds of the environment where the coats would be worn in as input. The sounds produced were therefore highly dependent on the current location of the wearer. On the other hand, for the same brief, P1 used beeping sounds meant to be completely independent of the wearer's environment.

4.2.7 User Control - Provided vs. Absent

This refers to whether sound designers envisage that the user or listeners can have some control over the design of the sound in question. Brief 2 explicitly asked the participants whether they would change a sound design depending on who was listening to it. P1 and P4 were quite reluctant to change design, while P2 and P3 made modifications. Even more explicitly, the provision of controls for the users were discussed by some of the sound designers when asked to creating a luxury version of the coat with the air sensor. Both P2 and P4 decided that the wearer would be able to change the sounds that the coat produced to the wearer's preference. Both would allow the user to select their own input sounds with P4 stating, *"completely different sound library"* and P2 stating *"they can somehow programme their own sounds into it"*.

4.3 Approach to Sound

4.3.1 Sound Provenance - Synthetic vs. Real

Throughout the study, the sound designers described on a number of occasions how they would produce the sounds they were ideating. P2 often started from synthetic tones, sometimes dissonant and a-rhythmical. She often started from the idea of a pure sine tone, for example using oscillators and LFOs to produce the sound of the shape in brief 1. This is in contrast to her approach when designing the sound of a brain to be listened to by people who are vegan. On this occasion, she wanted to work with very organic sounds, specifically using a contact mic in trees and bushes to get lots of natural sounds to use. P4 used a similar approach when ideating the "sigh" of a unhappy shape by describing how he wanted to use a squeak from a door hinge and decrease it in pitch.

4.3.2 Audio Effects - Processed vs. Unprocessed

When describing the sound of a brain for engineers, both P1 and P3 described using human voices as part of their sound design, P1 to give the impression of activity and P3 as a means of representing different brain functions. P1 stated they would employ EQ and consider reversing the audio so that the words were not fully comprehensible. P3 imagined more a conversation between the different functions and a central controller. The idea of processing audio was also present when P4 was considering a sound design for the air pollution sensor, using a bit-crusher to process the wearer's own music, so that the worse the air quality the more processed the listener's music would be. P2 also considers audio processing for this task, with the use of filtering a number broadband noise sources, and with the filter bandwidth mapped to the air quality.

4.3.3 Communication - Existing vs. Novel

This refers to whether sound designers decide to employ and harness the power of existing and established audio communication systems in their design, such as beeps, alarms and the voice of Siri, or they attempt to create completely novel ways of communicating with sound. As mentioned before, both P1 and P2 introduce existing auditory

displays in some of their designs, especially in Brief 3. P2, however, also ideates a novel system to communicate the air pollution quality based on noise filtering. P3 and P4 tend to create novel sound design rather than rely on existing paradigms.

4.3.4 Perceptual Issues - Considered vs. Disregarded

This refers to the degree with which sound designers consider perceptual issues already at the ideation stage. On occasion, our sound designers where concerned with how the sound they were ideating would be perceived within the environment, for example, if it could be masked of misinterpreted, or if it would be distracting. For example, this was a concern for P2 when asked if she would change her sound design for the air pollution coat in a stormy day. Her original idea was around filtered noise signals but during this task she stated, *"If you think about wind, and about broadband noise, there's not that much difference. So, you know, the sound would need to be distinguishable through that (wind)"*. This is also highlighted when she discusses the sound design being in a frequency range close to that of traffic, with the concern that traffic is something that is often *tuned out* by those who hear it. The other sound designers do not express a consideration for sounds being masked or misinterpreted by the listener.

5. DISCUSSION

As seen from the results in section 4, we have identified three main themes which include a number of dimensions. These highlight the different strategies and axes along which sound designers developed ideas to complete the briefs. They do not apply to every idea, brief or participant rather provide an initial framework that designers working in SID could explore when seeking a solution to their specific problem. When looking at Overarching Strategy, we found that there was great diversity among the participants in relation to whether they would develop an audio-visual construct or an audio-only construct to facilitate ideation. A link between visuals and audio was expected for the first brief, where a visual object is given to the designers to sound design, but P2, for example, did not construct any visual representation of a triangle and her design focused purely on the sound. There was also variety among participants in relation to whether they were comfortable having full creative autonomy. P1 referenced seeking guidance more than other participants, and also struggled to ideate as much as others solely on the basis of the given briefs. P3 and P4 appeared more comfortable with complete autonomy after initial guidance from the authors, whereas P2 was happy to take the direction given from the briefs and interpret in her own fashion without questioning her own autonomy. Under the heading of Design Strategies, the Mapping dimension showed a wide variety of approaches. P4 tended towards using very literal connections between objects and sound design. While P1 and P3 often used cultural associations as a source of inspiration for their sound designs. P2 oscillated between these two approaches throughout the task. We did not anticipate that some participants would focus on imagining a sound producing system, rather than the sound itself. This approach was used by P4 in both Brief 2 and 3 so that at times he was unable to describe how the final audio output would sound, but stated that he could not know beforehand, confident that if the imaginary system would incorporate all the data, it would produce the desired sonic output. P3 described a system for the air pollution coat which would link the air pollution sensor data to the breathing of the wearer. The system was triggered with each breath and a beads moving in a tube sound generated, the air pollution data linked to the number of beads and tube material. As we would expect, participants had a wide approach to sounds in relation to the dimension labelled Sound Provenance. At the opposite ends are P1 who, similarly to P3, tended to refer to existing natural sounds, and P2 who seem to use as as starting point always a synthesised sine wave. P4 made use of synthesised and natural recorded sounds, including Foley. It was common for our participants to initiate their designs by imagining a scene in which their sound design belonged to. Both Brief 1 and 3 give a basic outline of a scene - *walking happily along the road* and a journey map given which one would think naturally draws the participant along this dimensions. P2 only occasionally discusses aspects associate with a scene, however she is the only designer who expresses concern that their designs might be perceived incorrectly or masked by other sounds in context. The number of ideas identified as shown in Fig. 1 shows that P2 overall had the highest number while P1 had the lowest. When comparing the dimensions, it could be hypothesised there may be a link between the dimension of Guidance and the number of ideas generated. As P2 embraced creative freedom, she was able to ideate more, while P1 found difficult to ideate without external directions. We also observe that the greatest number of ideas were generate by P3 for Brief 2. P3 often focused on setting a scene before developing his sound design ideas. As no scene was hinted at in the second brief, we hypothesise that P3 had to ideate more to be able to come up with a final sound design idea. The study seems to confirm that sound designers are highly influenced by their most recent area of work. P1, a sound designer for radio, often felt the need to be guided by a script or a voice something he explicitly stated *"I mostly try to give a body to a voice, a spoken voice"*. P2, a sound designer who works across a wide range of media, was the most comfortable with receiving little direction and with experimenting with synthesised sound. P3, who now works in research within sonic interaction design for robots, linked his final sound design to a piston sound. Finally, P4, who now works as a researcher focusing on the sonification of data, focused, in Brief 2 and 3, on describing a sonification system rather than the sonic output.

6. CONCLUSIONS

We have presented the results of a study into the process of sound design ideation involving professional and former professional sound designers. We have found a number of common dimensions used within their strategies for ideat-

ing and producing solutions. We have highlighted how different sound designers move within these dimensions to progress with their design ideas. We have shown that the designers vary in number of ideas generated and hypothesise some reasons for these differences. We plan further studies with professional sound designers with the aim to enhance and expand our findings.

7. REFERENCES

[1] L. Zattra, N. Misdariis, F. Pecquet, N. Donin, D. Fierro, and S. D. Days, "Practices and Practitioners: Outcomes from the APDS project [Analyse des Pratiques du Design Sonore]," in *Sound Design Days*, 2019.

[2] S. Pauletto, "Invisible seams: the role of foley and voice postproduction recordings in the design of cinematic performances," in *Foundations in Sound Design for Linear Media*. Routledge, 2019, pp. 61–81.

[3] S. Delle Monache, N. Misdariis, and E. Ozcan, "Conceptualising sound-driven design: an exploratory discourse analysis," in *Creativity and Cognition*, 2021.

[4] P. Dourish, *Where the action is*. MIT press Cambridge, 2001.

[5] P. Susini, O. Houix, and N. Misdariis, "Sound design: an applied, experimental framework to study the perception of everyday sounds," *The New Soundtrack*, vol. 4, no. 2, pp. 103–121, Sep. 2014.

[6] M. Houel, A. Arun, A. Berg, A. Iop, A. Barahona-Rıos, and S. Pauletto, "Perception of emotions in knocking sounds: An evaluation study," in *17th Sound and Music Computing Conference, Online*, 2020.

[7] D. Rocchesso, *Explorations in sonic interaction design: SID; COST*. Logos, 2011.

[8] M. A. Boden, *The creative mind: Myths and mechanisms*. Routledge, 2004.

[9] D. A. Schön, *The reflective practitioner: How professionals think in action*. Routledge, 2017.

[10] E. A. Edmonds, A. Weakley, L. Candy, M. Fell, R. Knott, and S. Pauletto, "The studio as laboratory: Combining creative practice and digital technology research," *International Journal of Human-Computer Studies*, vol. 63, no. 4-5, pp. 452–481, 2005.

[11] N. Cross, *Design thinking: Understanding how designers think and work*. Berg, 2011.

[12] L. Hay, P. Cash, and S. McKilligan, "The future of design cognition analysis," *Design Science*, vol. 6, 2020.

[13] P. J. Cash, B. J. Hicks, and S. J. Culley, "A comparison of designer activity using core design situations in the laboratory and practice," *Design Studies*, vol. 34, no. 5, pp. 575–611, 2013.

[14] H. Sharp, Y. Dittrich, and C. R. De Souza, "The role of ethnographic studies in empirical software engineering," *IEEE Transactions on Software Engineering*, vol. 42, no. 8, pp. 786–804, 2016.

[15] D. M. Fetterman, *Ethnography: step-by-step*, ser. Applied social research methods series. SAGE, 2010, no. 17.

[16] K. A. Ericsson and H. A. Simon, *Protocol analysis: Verbal reports as data.* the MIT Press, 1984.

[17] K. A. Ericsson and H. A. Simon, "How to Study Thinking in Everyday Life: Contrasting Think-Aloud Protocols With Descriptions and Explanations of Thinking," *Mind, Culture, and Activity*, vol. 5, no. 3, pp. 178–186, Jul. 1998.

[18] D. W. Eccles and G. Arsal, "The think aloud method: what is it and how do I use it?" *Qualitative Research in Sport, Exercise and Health*, vol. 9, no. 4, Aug. 2017.

[19] S. D. Monache and D. Rocchesso, "Cooperative sound design: A protocol analysis," in *Proceedings of the Audio Mostly 2016*, 2016, pp. 154–161.

[20] R. Selfridge and S. Pauletto, "Investigating the sound design process," in *Nordic Sound and Music Computing*, 2021.

[21] M. Arvola, "Interaction designers' conceptions of design quality for interactive artifacts," 2010.

[22] T. Boren and J. Ramey, "Thinking aloud: reconciling theory and practice," *IEEE Transactions on Professional Communication*, vol. 43, no. 3, pp. 261–278, Sep. 2000.

[23] O. Alhadreti and P. Mayhew, "Rethinking Thinking Aloud: A Comparison of Three Think-Aloud Protocols," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[24] H. Platnerm and I. o. D. a. S. University, "Bootcamp Bootleg Booklet," 2018, published: Institute of Design at Stanford.

[25] V. Braun and V. Clarke, "Using thematic analysis in psychology. Qualitative research in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[26] P. Cash and M. Štorga, "Multifaceted assessment of ideation: using networks to link ideation and design activity," *Journal of Engineering Design*, vol. 26, no. 10-12, pp. 391–415, 2015.

[27] J. Wasiak, B. Hicks, L. Newnes, A. Dong, and L. Burrow, "Understanding engineering email: the development of a taxonomy for identifying and classifying engineering work," *Research in engineering design*, vol. 21, no. 1, p. 43, 2010.

[28] M. Gonçalves and P. Cash, "The life cycle of creative ideas: Towards a dual-process theory of ideation," *Design Studies*, vol. 72, p. 100988, 2021.

Spatial Audio Mixing in Virtual Reality

Anders Bargum, Oddur Ingi Kristjánsson, Simon Rostami Mosen and Stefania Serafin
Aalborg University Copenhagen
[abargu17, okrist17, srosta17]@student.aau.dk, sts@create.aau.dk

ABSTRACT

The development of Virtual Reality (VR) systems and multimodal simulations presents possibilities in spatial music mixing, be it in virtual spaces, for orchestral compositions, or for surround sound in film and music. In this paper, we present design aspects for mixing audio in VR. By the use of interaction design principles and an examination of related research, we create a framework from which a virtual spatial-audio mixing tool is implemented and coupled with a digital audio workstation (DAW). The tool is tested against a similar computer version to examine whether the sensory benefits and palpable spatial proportions of VR can improve the process of mixing 3D and binaural sound.

1. INTRODUCTION

While there has been a lot of investigation in the world of designing virtual reality (VR) applications for computer music, especially within the field of Virtual Music Instruments (VMIs) and New Interfaces for Musical Expression (NIME) [1], little focus has been on new ways of graphically representing mixing, mastering, and audio effects processing. Traditionally, music mixing consoles are divided into functional sections, which constitute different metaphors in a stereo context [2]. As an example, each track has a channel strip, used as the main way to adjust the volume, with slide potentiometers seen as a universal metaphor for amplitude. The panning potentiometer represents a track's placement and spatial position, mapping the left and right position of a knob to the left and right location of a sound. These universal metaphors have been the standard way of representing sound sources in a stereo field and have, thus, been implemented in music mixing interfaces in different Digital Audio Workstations (DAWs). However, their figurative representations are harder to map to 3D sound and one could, therefore, look at non-traditional paradigms like the 'stage metaphor' when representing spatial audio. In the stage metaphor, also called a 'virtual mixer' [3], the level and stereo position (and possibly other parameters) are modified using the position of a movable icon on a 2D or 3D image of a stage [4]. As seen in figure 1, each source in the 'stage metaphor' is graphically visualised in space. It, thus, affects and utilises human visual localisation cues such as the ventriloquism effect, which makes a person perceive the sound coming from the location determined by the human visual system [5]. Based on the idea of the 'virtual mixer' paradigm, this paper presents a design framework for visually representing spatial and binaural music mixing using the perceptual, visual, and spatial dimensions of VR. The paper will examine how VR can facilitate sonic interaction and if VR's inclusion of multidimensional space and free rotation/movement, can enable a composer to visually place, move and mix sound sources intuitively, in a 3D space. We implement a virtual environment (VE) that can be linked with the DAW 'Ableton Live' allowing musicians, producers etc. to intuitively, effectively and accurately sketch ideas for spatial mixing. The aim of this study thus is to create an immersive and creative environment that produces a set of spatial coordinates and objects, which can be transferred to a traditional DAW. The environment is tested against a similar computer version to investigate if the process of mixing 3D sound in VR actually can be an improvement to the producer, whereas the VE as a concept is evaluated using a focus group of experts.

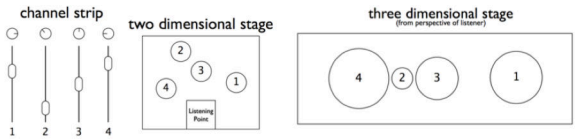


Figure 1. *Channel Strip vs. Stage Metaphor.* [6]

2. SONIC INTERACTION IN VR

There are multiple aspects to consider when designing virtual spaces and applications in VR, such as ensuring smooth interaction through minimum latency, preventing cybersickness, and facilitating presence. To fulfill smooth and successful interaction in computer music interfaces, Wang suggests that the system among other things should [7]:

- Be real-time if possible.
- Design sound and graphics in tandem and seek salient mappings.
- Hide technology and focus on substance.
- Introduce arbitrary constraints.

Copyright: © 2022 Anders Bargum, Oddur Ingi Kristjánsson, Simon Rostami Mosen and Stefania Serafin et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

This means that interaction with sound sources should be easy, quick, streamlined, and noticeable. Virtual objects need to match the location and motion of auditory objects, and the user should not be confronted with technology or implementation to increase excitement and interest. These aspects can additionally be supported using feedback and various studies state that especially haptic and tactile feedback allows a user to develop musical skills and understanding of controls [8]. Gelineck et al. as an example investigate the inclusion of external controls that allow for touch or vibrational feedback, by comparing the stage metaphor (in form of an iPad app visualising a stage) to the channel strip metaphor (normal faders and panning) when completing a stereo mix [9]. While the study does not find any significant difference in terms of performance between the two cases, the iPad application was preferred by the participants user experience-wise due to its "intuitiveness", "enjoyability" and its "ability to reveal the spaciousness of the mix" [9]. In terms of the effect of aesthetics, when interacting sonically in VR, Wang proposes different principles for graphical and visual representation of objects and environments in a 3D world [7]:

- Simplify: Identify core elements, trim the rest.
- Animate, create smoothness, imply motion: It is not just about how things look, but how they move.
- Be whimsical, organic: glow, flow, pulsate, breathe and imbue visual elements with personality.
- Aesthetic: have one; never be satisfied with 'functional'.

Gale et al. additionally suggest that one should avoid visual clutter, meaning too many objects potentially overlapping and/or occluding each other on the screen [10]. As a part of object cluster, Serafin et al. state that general control additionally can be enhanced by visually representing the player's body [8]. People cannot see their own body in VR and the frustration/confusion this may cause can be overcome by generating a visual substitution of a person's real body seen from first-person perspective [8]. All of these topics, be it feedback, real-time interaction, animation, or the 'virtual body ownership' elicited by representing a virtual body, can be facilitated by a VR system and there is no doubt that a VE thus has the potential of fulfilling successful sonic interaction.

Looking at the auditory facets of sonic interaction in VR and VEs, a main attribute is the space/room in which the sound is played. The sound itself will in different physical spaces be shaped by the room's spectral characteristics and modified by properties of the room. The perception of room acoustics is highly important for both the feeling of presence but also to elicit localisation capabilities and out-of-head localisations from a potential user [11]. One can choose different methods when employing models of spatialisation, acoustics, and reverberation to virtual rooms. Robert Hamilton distinguishes between two main models: the user-centric perspective and the space-centric perspective [12]. In the user-centric perspective, the sound will be manipulated from a 'first-person' point of view, where sounds in the virtual world will correspond to the real-world based model of hearing: they will be placed in a general aural spectrum known from every day, with corresponding depth cues implemented through filtering and delay components. This can be done by tracking the coordinate distance between event locations and the user's in-game avatar and matching given head-related transfer functions (HRTFs), or similar spatialisation algorithms, to the position of the sound [12]. The space-centric perspective, on the other hand, shifts the focus to the sound itself correlated between the virtual and physical world. In this model, sounds are no longer contextualised based on their proximity and relationship to a given user [12]. Instead, they are processed concerning both the virtual and physical world, meaning the placement in each environment will affect it. A spatialised speaker system allowing for multiple users and a communal experience is an example of this [12].

3. DESIGN OF VIRTUAL ENVIRONMENT

The conceptual idea of the virtual mixing environment is pictured in figure 2. As illustrated the user is placed in a 3D VR environment where different tracks from an arbitrary DAW, in the case of this project being *Ableton Live*, are represented as spherical sound sources in space with labels matching those of the DAW. A ray is cast from the controllers to signify which sound source is interacted with. The controller responds with vibration to signify contact between the ray and a sound source of choice. After selecting a sound source, the user can now move it in space from which data on position, distance, and angles to the head of the user will be collected. The data is passed into a binaural rendering system made in Max MSP [1], where the spatialisation is processed. This results in a match between visual and auditory locations of the sound sources and gives an audiovisual experience in space.

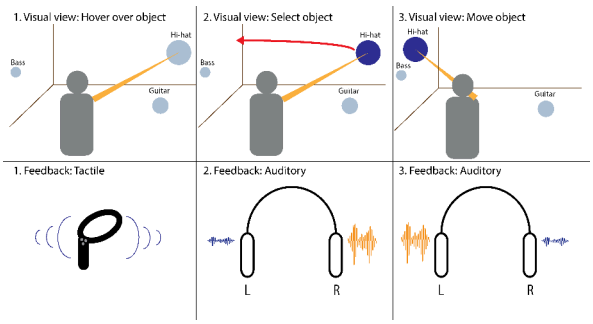


Figure 2. *Illustration of the concept.*

From the design principles presented earlier, the following decisions have been made for the virtual mixing environment:

1. A simple yet aesthetic environment will be created to focus on the importance of the mixing task. Com-

[1] https://cycling74.com/products/max

plex scenes such as concert halls and theater stages have been avoided as this might take focus away from listening.

2. The user-centric perspective will be utilised as it matches the 3D audio use-case and elicits the possibility of matching experienced sound to out-of-head localisation cues supporting the virtual body-ownership.

3. Spheres/sound sources are depicted around a stage to utilise benefits of the virtual mixer/stage metaphor such as 'intuitiveness', 'enjoyability' and 'ability to reveal the spaciousness of the mix', as earlier stated by Gelineck et. Al (section 2).

For the spatialisation of audio, dynamic binaural synthesis was implemented. For the synthesis, head-related impulse responses (HRIRs) from the MIT Media Lab [2] were used. The pack includes IRs ranging from -40 to +90 degrees on the vertical axis, where each elevation angle had corresponding IRs for 360 degrees on the azimuth in 5-degree intervals. The values of each IR were stored in text files readable as matrices in Max MSP and convolved with incoming audio tracks real-time, depending on data sent by the VE. Linear interpolation was implemented for each IR on the azimuth. The convolution of the incoming signal and the different HRIRs was done in frequency domain allowing for faster processing, whereas distance simulation and real-world spectral cues were simulated using the inverse-square law in conjunction with subtle coloration from low-pass filters.

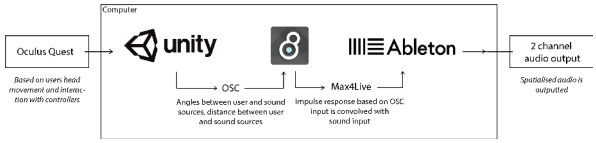


Figure 3. *Illustration of the system.*

The pipeline of the interactive VR environment was as follows:

1. Firstly, a combination of the Oculus Quest system and the game engine "Unity" was used to create a 3D environment that allowed the user to manipulate and position objects within a virtual space.

2. Secondly, object coordinates, angles, and user's head rotation, were sent through Open Sound Control (OSC) to Max MSP via the User Datagram Protocol (UDP) connection. The VE itself is thus not producing any audio output, rather it sends information to a DAW, which can be recorded for a binaural mixdown.

3. Max MSP and Ableton Live executed real-time sound rendering and binaural synthesis.

[2] `https://sound.media.mit.edu/resources/KEMAR.html`

An overview of the different stages, systems and the software used, can be seen in figure 3. The final VE used for evaluation purposes, is pictured below in figure [3] 4.

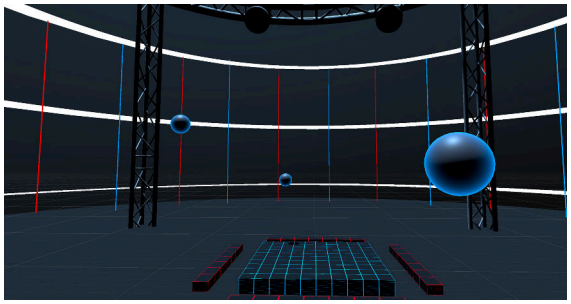


Figure 4. *Final design of the environment.*

4. METHODOLOGY

The VE and its communication of sound spatialisation was evaluated in two different ways. Firstly, mixing tasks with amateur music producers were carried out, assessing mixing precision and time spent on recreating a mix, pre-made using the developed spatial mixing tool. The mixing task was done comparatively, comparing the VE with the same environment implemented as a computer screen version e.g. the user would move icons on a screen with a mouse and keyboard rather than using a head-mounted display (HMD) and controllers. The computer version was meant to act as a test condition representing a traditional workflow that did not include the use of spatial interaction and perceptual cues. To quantitatively assess and isolate potential improvements of the VR version, differences in mix *precision* and general time *efficiency*, were measured. Precision was calculated based on relative distances between sound sources in a target mix and the participants' recreation thereof. Secondly, a focus group interview with music production students at the "Royal Rhythmic Academy of Music" in Copenhagen was carried out to qualitatively research general expert opinions on the product and its potential use-cases.

The bank of impulse responses contained 145 files for the azimuth, whereas 1434 files would be included for a full spherical binaural experience. As part of optimising the convolution process with the HRIRs, which is a computationally heavy process, and in order to simplify the information of the matrix containing the HRIR information, it therefore was examined whether elevation cues, in form of elevation HRIRs only, actually were needed from a perceptual perspective. The perceptual experiment was conducted (n = 14) at Aalborg University Copenhagen. The participants of the study were informed of the research question *"Do you feel like the sound is matching the position of the object?*" before the test started and asked to answer either 'yes' or 'no', with the option to hear the sound

[3] For video demonstration see `https://vimeo.com/674596681`

again. Hereafter, a series of visual 'sound sources' happening simultaneously as the spatialised auditory stimuli at different locations, either matching or not matching the visual sources, was presented to the participants. It was found that 95.2% felt the audio matched the visual position when the visual stimuli were elevated while the auditory stimuli still were kept at 0 degrees. On the other hand, 91.4% of the participants felt that the audio matched the position of the visual stimuli when both were at zero elevation. The results from this evaluation show that having visual cues for a given audio source, made the participants interpret the sound to originate from a visible object. Related research additionally shows that individual azimuth localisation is resistant to both elevation and reverberation [13], and it was therefore decided that HRIR convolution for vertical movement safely could be excluded in favor of processing power and general system complexity. The downside of this decision inevitably limits the systems potential and realism especially for sources placed at extreme vertical position where azimuth is irrelevant. However, only using azimuth IRs should be enough for a spatial audio mixing proof of concept.

5. PARTICIPANTS AND PROCEDURE

Both the focus group interview and the mixing task evaluation took place at Aalborg University in Copenhagen. 24 participants with different musical backgrounds and mixing experience, took part in the mixing task evaluation. Convenience sampling was utilised as all participants were part of the researchers' network. All the participants had experience mixing music where 54.2% of the participants had 3+ years of experience and the remaining had 1-2 years of experience. Regarding experience mixing spatial audio (binaural, surround, ambisonic) the majority (54.2%) had not tried it before. The majority of the participants were additionally familiar with VR (83.3%). The mixing evaluation asked the participants to recreate a mix and allowed them to switch back and forth between the target mix and their mix. Sessions lasted between 25-45 minutes for each participant. 12 students and a professor from the 'Music Production Bachelor' of the 'Rhythmic Conservatory of Copenhagen' additionally attended a focus group interview, after having tried out and played with the system. The interview lasted for 90 minutes, was transcribed, and coded into important tags and labels.

6. EVALUATION

6.1 Mixing Evaluation

The VR mixing tool was evaluated through a mixing task against a computer version with similar functionality in order to shed light on eventual differences in precision across the two media. In the computer version the user can simulate head rotation by right-clicking and dragging the mouse. A 'track' is 'grabbed' by hovering the mouse cursor over a 'track-object' and holding the left mouse button, whereafter it is possible to move and place the grabbed object using the keyboard keys 'W-A-S-D'. Both mouse actions can be performed simultaneously to allow for moving objects and rotating around the scene at the same time. The reference mix of the mixing task has been realized with the proposed VR system. The means and standard deviations of the collected data for both time (in seconds) and sums of relative precision compared to the reference mix (in Unity units) are shown below.

	Precision (Unity units)		Time (s)	
	Screen	**VR**	**Screen**	**VR**
Mean	35.74	35.60	558.38	448.04
Std. dev.	12.64	12.48	325.21	248.17

Table 1. *Mean and Std Deviation of Precision and Time across the two experimental conditions.*

QQ-plots and Anderson-Darling tests confirmed that the gathered *precision* data was normally distributed (p = 0.4381 for screen, p = 0.0693 for VR) while the *time* data was found not to be normally distributed (p = 0.0005 for screen and p = 0.0422 for VR). Since only the *precision* data were normally distributed (see figure 5), t-tests were used to test for the null-hypothesis *"mixes made by the participants in the VR version has no difference in, or less, relative precision to the reference mix, compared to the computer version"*. No significant difference was found between the means of the two conditions and the null hypothesis could thus not be rejected (p = 0.9531). However, due to dynamic and stereo errors in one of the audio clips used in the target mix, several participants stated that this specific audio track was exceptionally difficult to localize in both test conditions. If the position of this track is left out, the t-test rejects the null hypothesis (p = 0.0015). Thus it points towards a tendency of mixes made in the VR version having a higher relative precision to the reference mix, than the mixes made in the computer version as long as audio sources are dynamically static and mono.

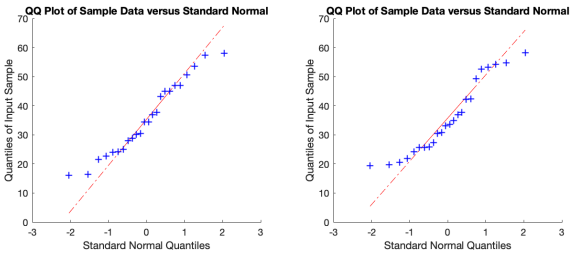


Figure 5. *Q-Q plots to check for normality of distribution of precision data. Left: screen version, right: VR version.*

6.2 Focus Group Interview

The second part of the evaluation consisted of a focus group interview and is below divided and categorised into themes and main topics, derived from a coding process. Table 2 highlights different quotes from the interview supporting the opinions on each topic.

6.2.1 Efficiency and Precision

The participants were asked about their initial thoughts regarding VR as a mixing tool and were all concerned of how efficient it potentially could be. Some participants felt that VR might be used as a quick sketching tool and they compared it to a big brush painting on a canvas. Besides the concept of this project, it was stated that it could be used as a more creative tool, rather than something one would use for precision. There was a general consensus that a DAW was expected to be more precise than the implemented VR program. It was mentioned by one of the participants that determining the program's efficiency and precision was difficult when they have not spent more time using the VR program, but that the program in general, together with the binaural algorithm, was experienced as something quick rather than precise. All participants agreed that the program gave enough information to make a judicious mix, but that the controllers made it hard to fiddle around and go into small details position-wise.

6.2.2 Spatial Sound Algorithm and Features

One participant described the panning as being "underdimensioned", though in general the spatial algorithm was found to be satisfying. A few participants noticed the exclusion of elevation, whereas most participants felt the match between sound and source movement realistic. Multiple participants described how they could imagine using this tool to create automation. Having different visual representations of the different sound sources as well as having a visual representation of sound activity on each track as a VU-meter on mixers was also mentioned.

6.2.3 Environment and Concept

It was stated that the virtual room could set a mood for the production by having different abstract elements. One participant pointed out that if the room should set a mood, it should be in a visually abstract way and not by looking realistic since this was the way they mixed mentally. It was, furthermore, stated that the decision of keeping the environment relatively neutral made sense in order not to influence the mix in an undesired direction and that the visuals used were pleasant and made sense in a mixing situation.

A participant pointed out that they found the prototype to be useless for them since it was designed for spatial mixing and not directly suitable for exporting a stereo mix. They pointed out that the prototype seemed to be designed for the producer to have a good experience instead of the final listener to have a good experience. It was mentioned by one participant that the application would be more relevant to use if it at least included the functions of a large-format console channel strip, for each audio track.

6.2.4 Comfort in VR

One participant explained how they felt dizzy after using the prototype, while another participant imagined that they could not spend more than 5-10 minutes in VR. It was furthermore discussed by several participants whether switching between headset and screen was better than staying in VR. Both ideas were supported by different participants.

7. DISCUSSION

While it is evident from the mixing task evaluation that recreating a mix in VR in some instances, and for some instruments, is more precise than doing it in an equivalent computer version, it is hard to draw any definite conclusions. Firstly, the mixing task evaluation was based on comparison, and it can be argued that the competition between the two conditions was slightly unfair due to the PC version being operated using a mouse and keyboard. Since the VE was designed to facilitate the spatial nature of VR, it could have been beneficial to compare the VR mixing tool against conventional spatial mixing tools for DAWs instead. Additionally, a quantitative measure in the form of *precision* was used to evaluate the mixing tool. While this allowed for quantitative comparison, including statistical hypothesis tests between interfaces, it is questionable whether the most *precise* tool is the best tool, as the task of mixing audio might be considered a subjective process and an artform. *Precision* as a valid evaluation metric in this instance, simply is debatable. The precision measure from the mixing task evaluation should therefore rather be used to support the qualitative data from the focus group interview, to establish a full picture. Lastly, it would have been desirable to have had multiple reference mixes, created on both conditions. This would have given a better understanding of navigation in the two environments and avoided any bias.

Concerning the focus group interview, two main findings are clear: 1. The participants saw the product as a quick sketching tool to test ideas and outline mixes rather than a tool to control precision and finer spatial details within the sound, and D. The participants were overall positive about the interaction with the product and its visual appearance, sensory benefits, and intuitive controls. In relation to the first finding, several participants stated that time and intuitiveness, in general, was an important aspect for them in a mixing tool/device and that the program indeed seemed to facilitate this. A participant expressed, among other things, that they "could imagine that you would get done faster with some things", and that the program seemed "very effective". Furthermore, participants agreed that the VR program definitely could be used as a quick sketching tool for swift ideas and testing of audio placement in a given space. This could, among other things, have been a result of the intuitive way of placing sound sources as well as the quick dynamic sound feedback and the possibility to link it up with Ableton Live. This could also have been due to the simplicity of the environment and the fact that only fundamental controls, pre-made audio effects, and interaction possibilities were included, giving it a 'to the bone' concept. Besides allowing positioning of sound sources (panning and volume), the program simply did not offer state-of-the-art possibilities such as the potential to manipulate the sound in finer detail, thus forcing the participant to use more time in the environment. This was moreover seen in the 'features' discussion of the interview, where participants emphasised a need for interactive dB meters, mute buttons, and the possibility to

Efficiency and Precision	”I could imagine that you would get done faster with some things. It seems very effective.”
Spatial Sound Algorithm and Features	”I found it slightly under-dimensioned so when you panned things to the side it was not as much as you would imagine. Front and back made good sense.”
	”It could also be used to do automation in a mix [...] You would have a much bigger area to draw on. I think that would be extremely useful.”
	”I think it is necessary to know that there is activity on the track”
Environment and Concept	”I think as the program is right now it might work even better for people who do not have experience making music and have to learn to visualise music in an extremely intuitive way.”
	”I cannot accept that I have not decided what it is this movement does. [...] I do not have any emotional connection to this.”
	”When I mix it is definitely something visual happening in front of me, I see the elements in front of me. It is not necessarily that I see the orchestra in front of me, it is much more abstract. A sprinkle over here, the sub-frequencies being another shape.”
Comfort in VR	”(In the environment, I could spend) 5-10 minutes or something like that”
	”I felt a bit sick. When I took off the glasses I felt really dizzy, but I think it is something you maybe have to get used to.”

Table 2. *Selected quotes from focus group interview.*

”do automation in a mix”.

It is worth discussing whether or not the HMD used for the project was the correct choice. The Oculus Quest was the chosen HMD due to it being wireless and thus consumer-relevant, providing the highest screen resolution compared to similar devices, as well as having a satisfactory refresh rate. However, since the hardware was built into the HMD, the computational power was limited. Limited computational power ultimately resulted in limited features in the final mixing tool. Features such as different shapes for different instruments and additional visual feedback were excluded from the implementation to accommodate low CPU load thus potentially reducing the overall usability of the tool. Additionally, the focus group agreed that adding more tracks in the VR environment would introduce clutter problems matching the suggestions in [10] in section 2. As only five tracks were part of the mix in the evaluation, having more could potentially eliminate the benefits of VR compared to PC. Related to this, it was stated: ”When we tried it here it was very manageable with five tracks, but if you have 67 tracks [...] it might hinder you more than it helps.” A suggestion for this was being able to group tracks. Concerning the concept, some participants struggled to grasp the core idea behind the product, dynamic spatial mixing. The fact that the mix changed relative to head movement confused many participants and hindered them in understanding the possibilities and functionality of it. As one participant said, ”I often ended up looking one way and then imagining that I mixed in stereo [...] this just made me feel that everything was imprecise.”

8. CONCLUSION

This paper has presented the design, implementation, and evaluation of a VR environment controlling dynamic binaural synthesis for 3D audio mixing. A real-time Max MSP patch was implemented to convolve incoming audio with HRIRs retrieved from data sent by the VE through OSC. The implementation allows for real-time sound rendering and binaural synthesis based on virtual sound location data. Both qualitative and quantitative evaluations were conducted through the form of a focus group interview and a mixing task evaluation. The result from the mixing evaluation hints toward the VR mixing tool improving mix precision for a simple audio mix, compared to a computer version. The answers from the focus group interview, furthermore, confirm the potential of a VR mixing tool and its spatial as well perceptual benefits. However, it is stated that it could better serve as a creative ’sketching’ tool to quickly try out different ideas than a precise spatial audio mixing environment, due to the lack of fine adjustment possibilities and potential visual clutter.

Acknowledgments

The participation to the conference was supported by the European Art Science and Technology Network (EASTN-DC).

9. REFERENCES

[1] A. R. Jensenius and M. J. Lyons, *A NIME Reader: Fifteen Years of New Interfaces for Musical Expression*. Springer, 2017, vol. 3.

[2] S. Gelineck, M. Büchert, and J. Andersen, “Towards a more flexible and creative music mixing interface,” 04 2013, pp. 733–738.

[3] D. Gibson and G. Petersen, *The Art of Mixing: A Visual Guide to Recording, Engineering, and Production*, ser. Mix pro audio series. MixBooks, 1997. [Online]. Available: https://books.google.dk/books?id=T34yQAAACAAJ

[4] B. De Man, N. Jillings, and R. Stables, “Comparing stage metaphor interfaces as a controller for stereo position and level,” 09 2018.

[5] S. Yantis and A. A. Richard, *Sensation and Perception*. New York, NY: Worth Publishers, 2014.

[6] J. Ratcliffe, "Hand and finger motion-controlled audio mixing interface," in *NIME*, 2014.

[7] G. Wang, "Principles of visual design for computer music," 09 2014.

[8] S. Serafin, C. Erkut, J. Kojs, N. Nilsson, and R. Nordahl, "Virtual reality musical instruments: State of the art, design principles, and future directions," *Computer Music Journal*, vol. 40, no. 3, pp. 22–40, 2016.

[9] S. Gelineck, D. Korsgaard, and M. Büchert, "Stage- vs. channel-strip metaphor: Comparing performance when adjusting volume and panning of a single channel in a stereo mix," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME 2015)*, E. Berdahl, Ed. Louisiana State University, 6 2015, pp. 343–346.

[10] W. Gale and J. Wakefield, "Investigating the use of virtual reality to solve the underlying problems with the 3d stage paradigm."

[11] U. Zölzer, *DAFX: Digital Audio Effects*. Wiley, 2011. [Online]. Available: https://books.google.dk/books?id=jILxqgnjDKgC

[12] R. Hamilton, "Building interactive networked musical environments using q3osc," 02 2009.

[13] E. Méaux and S. Marchand, "Sound Source Localization from Interaural Cues: Estimation of the Azimuth and Effect of the Elevation," in *Forum Acusticum*, ser. Proceedings Forum Acousticum, Lyon (on line), France, Dec. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03042326

Issues of Ubimus Archaeology: Reconstructing Risset's Music

Victor Lazzarini
Department of Music,
Maynooth University, Ireland
victor.lazzarini@mu.ie

Damián Keller
Amazon Centre for Music Research
Federal University of Acre
dkeller@ccrma.stanford.edu

Nemanja Radivojević
University of Bern
Hochschule der Künste Bern
nemanja.radivojevic@hkb.bfh.ch

ABSTRACT

This paper discusses questions of ubiquitous music archaeology within the context of reconstructing Jean-Claude Risset's music with MUSIC V. We begin by situating the field of ubimus archaeology as it overlaps with the area of software archaeology. We then explore the theory, where issues of sustainability, archival, reproduction, and recovery come to the fore. The next sections apply these ideas to a specific case, that of the reconstruction of computer music from original sources. We look at the work involved in restoring MUSIC V from its original Fortran code, and the attempt to bring it to a state where the music of Risset can be reconstructed in a modern computing environment. This involved extensive archival work to rescue and reassemble his original score files. The paper concludes with an overview of the current state of this project, some preliminary results and the next steps.

1. INTRODUCTION

Whether we look at the rate of change of computing power or at the way we conceptualise music making, over the span of six decades the design of computing tools has impacted how we create and how we experience music. The rapid changes of techniques and of the material support for planning, execution and sharing of technological resources sometimes obscures our vision of the state of the art of musical endeavours on the one hand, and of the extant knowledge inherited by current practises on the other. Take as an example the emerging field of ubiquitous music (ubimus) [1]. The emphasis of some ubimus approaches on the development and enthusiastic adoption of innovative design techniques could be misread as a dismissal of the previously acquired knowledge through a long legacy of music practises. This interpretation is wrong. To prove this point, Lazzarini and Keller [2] report the results of three "archaeological excavations" of the original versions of historically significant music-programming environments, MUSIC V, cmusic, and Csound. They label this endeavour *ubimus archaeology*.

Ubimus archaeology is conceptualised as a multi-stranded initiative that combines current technical know-how with applications that expand the frontiers of musical archaeology. Musical archaeology aims to increase our understanding of the past through the study of the material traces left by musical activity [3]. For music practises centred on the use of acoustic instruments, there is a fairly stable material base and a set of well-established methods to document and select the relevant information. As music-making starts to incorporate technological resources, it increasingly becomes dependent on the availability and maintenance of specific hardware and software. Consequently, music becomes as disposable as the resources enabling its existence. Some practises can be partially recovered through the documentation of their sonic products. But more often than not, the knowledge gained through the creative processes and through the community-shared musical experiences is lost. These aspects are key targets of ubimus archaeology.

In this paper we report work done to implement a functional replica of the MUSIC V acoustic compiler. This is a significant expansion of the work presented by Lazzarini and Keller [2], including the addition of tools to streamline the usage of the software and major adaptations to meet the requirements of current computer-music standards. Furthermore, we report the application of such work to the reconstruction of early works of computer music by Jean-Claude Risset. In line with ubimus archaeological objectives, we chose four targets that the artistic community considers worthwhile sonic and musical products of the past: *Little Boy*, *Mutations,* the *Introductory Catalogue of Computer Synthesized Sounds* and the *Computer Study on Trumpet Tones*, whose sources have been made available by the Fonds Risset of the PRISM laboratory. We include examples of the original files and furnish a description of the stages necessary to achieve successful replicas of creative processes based on technology previously thought to be extinct.

To frame our results, we situate the ubimus archaeology initiative within the context of ongoing efforts to increase the sustainability of both technological and musical practises. We also discuss emerging issues related to replicability and community-shared knowledge in an effort to expand the life cycle of creativity-support resources. Then we tackle the procedures of recovering MUSIC V from the ashes and point to the research avenues opened by its working replica. As an applied case, we describe the materials and procedures used in the partial reconstructions of the creative processes adopted by Risset in his projects from the sixties. We also include pointers to undocumented and original materials that may prove to be useful for future archaeological endeavours targeting Risset`s work. After a summary of results, we discuss the implications of this study for

ubimus archaeology and information-technology design with an eye on potential future applications.`

2. ARCHAEOLOGICAL ENDEAVOURS AND RELATED WORKS

The widespread adoption of computational resources for music making and the increased presence of information technology in our daily life have opened both opportunities for innovative practices and an enormous can of worms of socio-cultural issues. Two aspects that have long-lasting consequences are the sustainability of the deployed infrastructure and the replicability of the procedures and products linked to the musical activity.

Replicability in music is doubly problematic. On the one hand, the standard methods of scientific inquiry have been expanded by the initiatives grouped under the rubric of practice-based research. This expansion changes the meaning of musical research and highlights the specific contributions of artists [4]. Artistic replicability may demand very stringent conditions that are not aligned with the expectation to obtain an identical product when applying the same procedures. Creativity-oriented support should foster diversity rather than mechanical replication. On the other hand, the dimensions of musical experience are not limited to the acoustic properties of sound. Yes, being able to freely and easily replicate a sonic product without introducing errors or degrading its quality is an important contribution of digital technology. But music-making activities involve a network of human and non-human resources with volatile symbolic associations that are difficult to understand just by analysing their sonic outcomes. Thus, replicating a musical experience not only entails recovering its material resources, it also involves engaging with the dynamic properties that emerge *during* the activity. As it will become clear when we discuss the methods employed in this project, these two layers of replicability involve complementary strategies.

Sustainability has emerged as a key aspect of technologically oriented musical endeavours [5, 6, 7]. Sustainable practices highlight the importance of a persistent material base for musical activities, including not only the tools needed to obtain sonic products but also the know-how inherited and produced throughout the technological design cycle. During the initial years of music-computing, the only places where it was possible to conceive and carry out artistic projects were the large research centres of universities and information-technology companies [8]. This restriction involved an infrastructure that demanded serious efforts of implementation. But on the positive side, over its first three decades of existence the infrastructure remained fairly stable. Today, the situation is completely different. We witness fast changes and widespread access to short-lived hardware, compounded with fast circulation of tools and know-how enabled by the internet. Thus, the tendency of software to become obsolete or unsupported has suffered a drastic increase.

How can we deal with the menace of disposable technology? Ubimus archaeology may open a window to the past that could furnish useful information on both the computational and the artistic strategies applied before, during and after the execution of a musical project. Some of these strategies may involve design choices that make sense at the specific historical context but over the years become conceptual culs-de-sac. Other strategies may survive as legacy approaches which linger on despite their drawbacks. And finally some of the early designs may prove their applicability in various contexts despite the sharp differences in social and aesthetic expectations between historical and current music practises.

3. MUSIC V

This section examines the recovery of a classic piece of software, the MUSIC V acoustic compiler, which was used by Risset to compose his works from the late 1960s onwards. It was written by Max Mathews, Joan Miller and others at Bell Labs using Fortran as the implementation language [9]. Although it was not the first of its kind written in a high-level language (as it was preceded by MUSIC IVBF from Princeton), it was widely shared and used in various places in the USA and Europe in the fifteen or twenty years following its creation. It is perhaps for this reason that we are actually able to recover it from sources and attempt to bring it to an operational level that allows us to reconstruct some of the music composed through its use.

3.1 System Structure and Operation

The structure and operation of MUSIC V follows more or less the design introduced by MUSIC IV. It takes in a score prepared by the user (the source code for the composition written using the MUSIC V language) and outputs a sound file. Originally, input was supplied in the form of punched cards and output was written to digital computer tape. There are three stages, or passes, involved in the process:

- Pass 1: parses the score, producing a numerical representation of it as output, optionally applying a Fortran PLF routine to generate or process data.
- Pass 2: takes the pass 1 output, applies action time sorting of all data statements, time scaling, conversion of data (optional Fortran CONVT routine supplied by the user), and optionally applies a supplied Fortran PLS routine.
- Pass 3: takes the sorted, time-scaled, converted numeric data and synthesises the sound.

From a modern perspective, we can think of these as three separate programs, taking different types of inputs and producing different types of outputs. While the original operation used punched cards for the text and numeric inputs/outputs, in today's operating systems, plain text files are employed and the programs can be run as terminal commands.

3.2 Restoration

The work to restore MUSIC V is based on three main sources:

1) The Fortran source code, as typed and reorganised by Bill Schottstaedt in 2008.
2) Max Mathew's MUSIC V manual and examples from his book [10].
3) Risset's *Catalogue of Computer Synthesized Sounds* [11].

Bill Schottstaedt's code was essential since it provided a means to re-build the software from a base that is very close to the original Fortran code. While we could potentially recreate MUSIC V from the other sources using a more current programming language, that would be a much more complex and error-prone project. However, it is also important to note that the source code has been modernised substantially (and it is therefore not exactly the original from 1970). In Schottstaedt's source code comments, it is possible to see that there was what was described as "major surgery" applied to the code in order to make the programs work again in a modern system with the latest Fortran compiler versions. His choice of leaving the original code commented out was very helpful as we could study how the software was originally conceived.

The provenance of the code is somewhat uncertain, but the comments indicate that it was

> "typed in from old XGP output[,] file last written 19Jun75 MUSIC5[M5,GM] (I believe GM = George McKee)[,] SAIL" [12].

Later on we see mentions of PDP in the comments, leaving us with a reasonable guess that this was the version kept at the Stanford Artificial Intelligence Laboratory, which had a computer of that line. The code was most likely kept as a deck of punched cards and someone in 1975 printed it in a Xerox Graphic Printer. We do not have any other direct indication of the earlier history of this code and so we can only assume it is reasonably close to the one used at Bell Labs.

It is likely that other printed sources may be available, but as of now this appears to be the only usable source in a machine-readable format. We have also seen various copies of the source files containing this code in Risset's hard drives kept by the Fonds Risset, but no other versions, although there were compiled Motorola 68000/20 binaries for the Apple Macintosh[1] amongst his electronic materials. Besides these, printed sheets from 1968/9 containing the source code were discovered (as reported later in this paper), but have not been examined in close detail yet.

Schottstaedt's version, while it could be compiled in gfortran at the time of its creation, was fairly incomplete. It ran some basic score examples from Mathews' book but was not capable of, for instance, rendering Risset's Catalogue. The FLT unit generator was also missing. In 2009, a version with some fixes and additions was produced and subsequently made available as part of the Audio Programming Book [13]. This had the additions of several GEN routines from Risset's Catalogue, as well as the general CONVT routine listed there. In addition, the IOS unit generator required for his endless glissando example (#513) was added as new code (with the assumption that it was an oscillator using a linear-interpolation table lookup). A shell script handling the copying of user score inputs to the score file read by the pass 1 program and copying the snd.raw output from pass 3 into a user-provided file name was also added to make running MUSIC V more straightforward. However, this version was also largely untested, and a bug in the code was preventing the CONVT routine from being called. Therefore, in order to run the scores from Risset's *Catalogue* some editing was needed to apply the necessary conversions.

A renewed effort in the last year has brought us somewhat closer to restoring this code to the point where the music of Risset may be reconstructed from his original scores. The following issues have been tackled:

- Further work was needed in resolving Fortran issues still left over, which caused errors and warnings. The code now is up-to-date with the latest version of the language.
- The bug preventing CONVT from running was fixed. This was in fact a simple typographical error where a variable name was typed in as a literal. However, any CONVT functionality related to frequency conversion was still broken because the sampling rate had been hardcoded in the sources (more on this below).
- The code could not run stereo-output scores. While the code for the STR unit generator was present, it contained a bug affecting the sample count. This was easily fixed. However, the number of output channels was also hardcoded to one in the sources.
- Supporting stereo required further work, or "major surgery" as described earlier. The output buffer needed to be increased in size to accommodate the data. The actual buffering mechanism needed to be debugged for this to be achieved. Following the logic of code written in 1970 is not exactly straightforward, but after many iterations a solution was found. However, the number of channels still needed to be hardcoded in order for it to work.
- Both the sampling rate and the stereo/mono switch are expected to be adjustable from the score. This is done via a mechanism that sets integers in all passes to indicate the required parameters. This mechanism was broken and it needed fixing. With this restored, a score containing STR automatically sets the output to stereo, and the SIA unit generator can be used to set the sampling rate in the score. An addition to pass 3 makes it write a text file documenting the

[1] M5Mac by Simone Bettini, written as part of his degree thesis, under the supervision of Giovanni de Poli at Padova. According to Betttini, this was a recreation, using newly written code, although based on the original program.

sampling rate and number of channels in the output file.
- The MUL unit generator was not producing any output and was brought back to operation.
- The pass 2 CON routine, which applies timescaling to the data statements, was fully broken in Schottstaedt's version. This was restored to the original code (which was commented out) with some adjustments to avoid numeric issues.
- The application of PLF routines in the first pass requires some adaptation due to the way Schottstaedt has reorganised the program and eliminated the original reading code (the READ1 Fortran routine). The PLF3 routine from example #500 in the *Catalogue* has been incorporated as an example of such adaptation work.

In addition to the restoration work, a number of enhancements to the code base have been implemented. A dependency-free sound file converter has been written in C to take the pass 3 output and produce a RIFF-Wave sound file. To facilitate operation, a driver program, also written in C, now takes the score input and produces a RIFF-Wave sound file by calling the three passes and the converter. A CMake project was added to simplify cross-platform building of the software. Finally, a code repository was created in github to share the development work as an open-source project [14].

4. RISSET'S SCORES

In this section we report on the findings in the Fonds Risset [Fonds Jean-Claude Risset, Laboratoire PRISM (UMR 7061 – France)], which enabled us to consider the reconstruction of Risset's music from original score codes. Risset's compositions and research work from the 60's and 70's are abundantly documented not only with sketches, drafts and written descriptions on compositional procedures but also with MUSIC IV and V code. Although pieces like *Inharmonique (*W21_001), *Mirages* (W23_001)*,* and *Songes* (W24_001) are all relatively well documented with preserved score code[2], we have opted to concentrate on the pieces and research from Risset's early period, *Computer Study for Trumpet Tones*, *Little Boy*, *Mutations*, and *Introductory Catalogue of Computer Synthesized Sounds*.

These works belong to Risset's first two visits to Bell Labs in 1964-65 and 1967-69. They share interests for certain problems, especially the questions of timbre and musical paradoxes. They belong to a period where, according to the available documents in Fonds, Risset was parallelly, on one side, learning and exploring the possibilities of digital synthesis and, on the other side, composing. This is suggested by numerous comments in score code that relate to the sound processing techniques as well as compositional issues (i.e., Fonds Risset, W20_004_1, p.4, 5 or R21_001).

Despite the fact that the *Study of Trumpet Tones* was done in 1964/65 in Music IV and the other three works in Music V, they all share similar computer instrument designs, compositional procedures and sound results. Additionally, we suspect that the source code might have been modified in the seventies when MUSIC V has been transferred to LMA in Marseille and Ircam, which could make our reconstruction of later pieces more problematic.

While the *Catalogue* has been published in its MUSIC V score code form, *Little Boy* and *Mutations* were never publicly available as score code, but only as rendered and mixed audio. Some researchers published parts of this code in their analyses [15, 16], but there was no such effort of a complete analysis for these early pieces as was done for *Inharmonique* [17]. More recently, Antonio de Sousa Dias adapted parts of *Inharmonique* and the *Catalogue* to Csound and Max [18], and Svidzinski and Tiffon recreated parts of the code of *Songes* using Faust [19]. However, to our knowledge, no previous reconstruction work has been attempted using MUSIC V.

Through our attempt to reconstruct the score code, we hope not only to shed some light to the genesis of classic works of early computer music but also to provide an insight on Risset's creative process of numerous forth and back movements from research to composition to develop the *Computer Study*, *Catalogue*, *Mutations* and *Little Boy.*

4.1 Fonds Jean-Claude Risset

Two years after Jean-Claude Risset's death in 2016, his entire estate was deposited in the Fonds Risset of the PRISM laboratory, part of the CNRS in Marseille, today under the direction of Dr. Vincent Tiffon. In the last two years we have collaborated on digitisation and cataloguing of an important part (including Risset's original MUSIC IV and V code prints, score manuscripts and sketches) of the material, which consists of well over 30,000 pages. This material in the form of JPEG files has been made available online to researchers upon request. Risset's notebooks, correspondence, concert materials and other documents of his compositional and research activity are (still) only accessible on site.

4.2 *Little Boy* Sources

The documents related to *Little Boy* are organised in numerous call numbers. Besides the manuscript, sketches and drafts of the orchestra version[3] [W10_05, W20_003_1], the manuscript of Naka [W10_007], a notebook with ideas and descriptions of sounds for *Little Boy* (and other pieces) [W20_003_1], press clipping for the theatre play *Little Boy* [Halet P. *Little Boy.* Directed by Guy Lauzin. Maison de la Culture de Nevers, 1974], correspondence with P. Halet [W20_003_3], there is a

[2] *Mirages* (1978) with its more than eight hundred pages of MUSIC V score code synthesised at IRCAM presents a valuable source for comparative research with the pieces made ten years earlier at the Bell Labs.

[3] *Musique pour "Little Boy" de Pierre Halet, épisodes orchestraux* [2222.2220.timp.3perc, soprano, e-guit., hp., pno./cel., string quintet, mixed choir] dated to 1968, 54 pages.

folder with printouts of the MUSIC V score code for *Little Boy* [W20_003_2]. Its 225 pages contain the score code on perforated continuous paper for line printer of the IBM 7094 computer, annotated by Risset with details on sampling rates, magnetic tape numbers, dates, code errors and/or possible solutions, instrument schemata, chords or notes to be played, ideas on sound to be produced, and possible technical solutions for certain passages; spectrograms of unidentified sounds annotated by Risset; and separated or clipped paper sheets with Risset's comments on the mix, instruments, code or other (including traditional notation examples, technical drawings of instruments and code examples).

4.3 *Mutations* Sources

The material concerning *Mutations I* is situated in the folder with the call number W20_004, and its five containing folders. Two of them contain score code printed as for *Little Boy* on perforated continuous paper for line printer IBM 7094 [W20_004_1, W20_004_5], as well as separated and clipped paper sheets, annotated by Risset with details similar to *Little Boy*. In total they contain 341 pages.

All the documents of W20_004_1 were organised by Risset himself in the following subfolders: Brassy; LB type episodes; Mutations - *description des runs*, Mutations - *descriptions des mixages*; Nonlinear transfer Db Sine etc.; Drums - Gong harmony - Mutations; Parallel glissandi - Mutations; Fonctions sérielles; and Recent runs - Mutations. Besides these materials, W20_004 contains drafts and autographs of the program notes [W20_004_2], a photocopy of "Interview with JCR" [20], the transcription of Mutations in standard music notation [21] [W20_004_3], a photocopy of a part of Lévy's book related to Mutations [15] [W20_004_4].

We also took into account call numbers with *Mutations II* (1973), which contain further score code from *Mutations I*. In addition to the parts synthesised in Bell Labs, *Mutations II* was run in the Institut d'Electronique Fondamentale d'Orsay, and it shares the same score code used in *Mutations I*. In addition to the sketches, drafts and the autograph for ensemble (Fl., Cl., perc., pno.) [W20_005_2, W20_005_3], handwritten notes on the piece, sketches of program notes, the folder W20_005_1 contains the score code MUSIC V.

4.4 *Catalogue* Sources

Since it was used by Risset on repeated occasions, *Catalogue* source code is scattered in many different folders. We haven't identified drafts and sketches made specifically for the *Catalogue*. Some of the code was found together with *Little Boy* and *Mutations* material. Some other runs seem to be originating from Risset's study on pitch paradoxes [22]. Folders R21_001 to R21_016 contain the final version in manuscript and print.

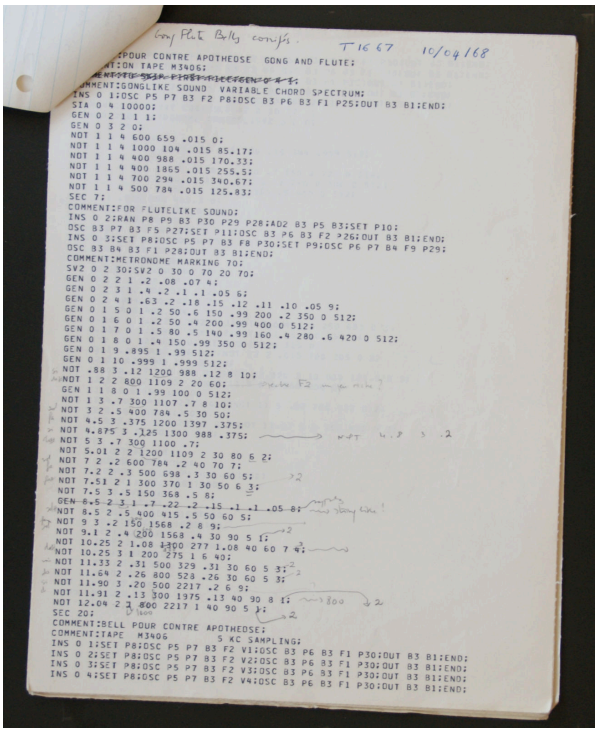
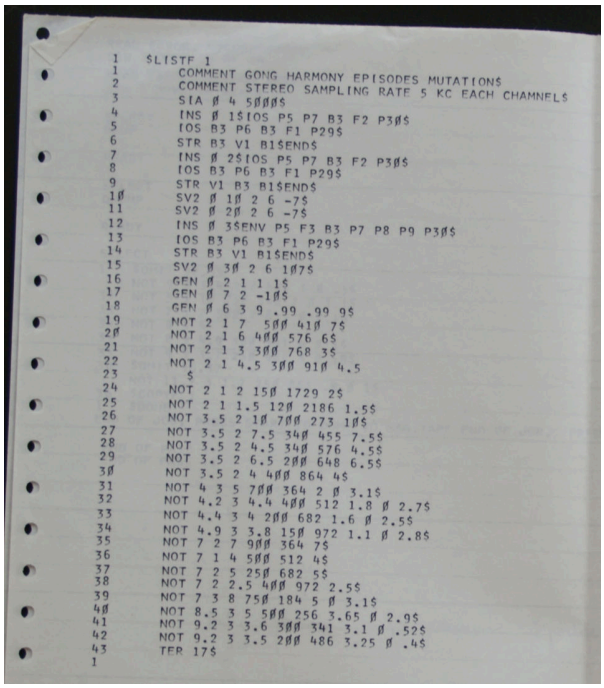


Figure 1. A page of the score code for *Little Boy*, W20_003_2, p.61.

Here we found a notebook with Risset's working notes (1967) [R21_001], a copy of *Music V: Quick Reference Manual* by F. R. Moore, 1967/68, a printout of Music V source code[4], annotated by Risset, 1968/69 [R21_002], score code, questionnaire and study design drafts for the *Pitch Study,* runs for glissandi for *Little Boy*; 1968/69, 284 pages [R21_003], annotated code for Shepard Glissandi [R21_007], a copy of Music IV manual [23][R21_008]

The folders R21_010 to 016 contain more specific material from the *Catalogue*: manuscripts and original printouts, two versions of *Catalogue* score code annotated by Risset, separate and clipped paper sheets with annotations on tape numbers, instrument schemata, accompanying description text manuscript (draft) of the *Catalogue*, letters to the publishers, distribution lists, reviews, James Beauchamp review article [24], Steven Held's translation of the *Catalogue* for MUSIC 11 (1990), a CD with text files from the *Catalogue* and Antonio de Sousa Dias' portings of some examples in Csound and Max [18], drafts of the texts for the Wergo CD [25], original *Catalogue* vinyl record 33+⅓ rpm [26]. Although it seems that much of this material is in its final form, there are several drafts of the code (especially in the folders R21_010 and 012) that might give us further insight into the genesis of the *Catalogue*.

[4] The discovery of this material will also contribute to further adjustments to our restoration of MUSIC V.



```
1   $LISTF 1
1       COMMENT GONG HARMONY EPISODES MUTATIONS
2       COMMENT STEREO SAMPLING RATE 5 KC EACH CHAMNELS
3       SIA ø 4 5øøø$
4       INS ø 1$IOS P5 P7 B3 F2 P3ø$
5       IOS B3 P6 B3 F1 P29$
6       STR B3 V1 B1$END$
7       INS ø 2$IOS P5 P7 B3 F2 P3ø$
8       IOS B3 P6 B3 F1 P29$
9       STR V1 B3 B1$END$
1ø      SV2 ø 1ø 2 6 -7$
11      SV2 ø 2ø 2 6 -7$
12      INS ø 3$ENV P5 F3 B3 P7 P8 P9 P3ø$
13      IOS B3 P6 B3 F1 P29$
14      STR B3 V1 B1$END$
15      SV2 ø 3ø 2 6 1ø7$
16      GEN ø 2 1 1 1$
17      GEN ø 7 2 -1ø$
18      GEN ø 6 3 9 .99 .99 9$
19      NOT 2 1 7  5øø 41ø 7$
2ø      NOT 2 1 6 4øø 576 6$
21      NOT 2 1 3 3øø 768 3$
22      NOT 2 1 4.5 3øø 91ø 4.5
23        $
24      NOT 2 1 2 15ø 1729 2$
25      NOT 2 1 1.5 12ø 2186 1.5$
26      NOT 3.5 2 1ø 7øø 273 1ø$
27      NOT 3.5 2 7.5 34ø 455 7.5$
28      NOT 3.5 2 4.5 34ø 576 4.5$
29      NOT 3.5 2 6.5 2øø 648 6.5$
3ø      NOT 3.5 2 4 4øø 864 4$
31      NOT 4 3 5 7øø 364 2 ø 3.1$
32      NOT 4.2 3 4.4 4øø 512 1.8 ø 2.7$
33      NOT 4.4 3 4 2øø 682 1.6 ø 2.5$
34      NOT 4.9 3 3.8 15ø 972 1.1 ø 2.8$
35      NOT 7 2 7 9øø 364 7$
36      NOT 7 1 4 5øø 512 4$
37      NOT 7 2 5 25ø 682 5$
38      NOT 7 2 2.5 4øø 972 2.5$
39      NOT 7 3 8 75ø 184 5 ø 3.1$
4ø      NOT 8.5 3 5 5øø 256 3.65 ø 2.9$
41      NOT 9.2 3 3.6 3øø 341 3.1 ø .52$
42      NOT 9.2 3 3.5 2øø 486 3.25 ø .4$
43      TER 17$
1
```

Figure 2. A page of the score code for *Mutations I*, W20_004_1, p.150.

4.5 *Computer Study of Trumpet Tones* Sources

Call numbers R20_ 001 to _016 (1867 pages) contain among others: typewritten *Computer Study of Trumpet Tones,* PISA tests of recorded trumpet sounds, score code for sound examples for the Trumpet Study, score code entitled Prelude, Study I-XV, Microtone Study I-III, Psycho, Purcell, Breize ma bro, etc., other unidentified score code, computer reports of executed runs, sketches and drafts of the study design, etc. We haven't located the audio examples for the *Computer Study*. Additionally the call number R21_017 includes printed and annotated spectrograms of trumpet tones for the study.

It is hard to distinguish Risset's composition from his research. Titles like *Prelude, Study I-XV, Microtone Study I-III,* or *Purcell, Breize ma bro*, are to be found often in the same folder indicating that Risset was experimenting with the synthesis parallely for his *Computer Study* of *Trumpet Tones* and his compositional activity. While *Purcell* and *Breize ma bro* clearly belong to the sound examples of the *Trumpet Study*, the *Microtone Study* represents an interesting discovery that is not easy to classify [27]. Although Risset later discarded all the computer pieces from this period [16], the preserved score code gives us the opportunity to analyse both his creative processes and research output.

4.6 Other Sources

Call numbers R24_001 to _003 and R25_001 to _016 contain more score codes from 1964-1969, but to date we have not done detailed research on this material. Additionally, numerous audio support (DAT tapes, CDs, audiotapes, magnetic tapes) are in the Fonds, but this material is not classified. The original tapes of *Little Boy* and *Mutations* are supposedly in INA, Paris [28].

4.7 Preliminary Results and Further Perspectives

With the thoroughly examined content of the Fonds Risset we have set the foundations for our research. Although it is plausible that some additional documents will be found in the future, we suppose that most of the surviving score code for the four works in question is located. Since no code was published as Urtext, at this point we do not know what code is the "autograph" and what is a sketch, study or draft. Once our reconstructions are done, we assume we will be a step closer to answering this question.

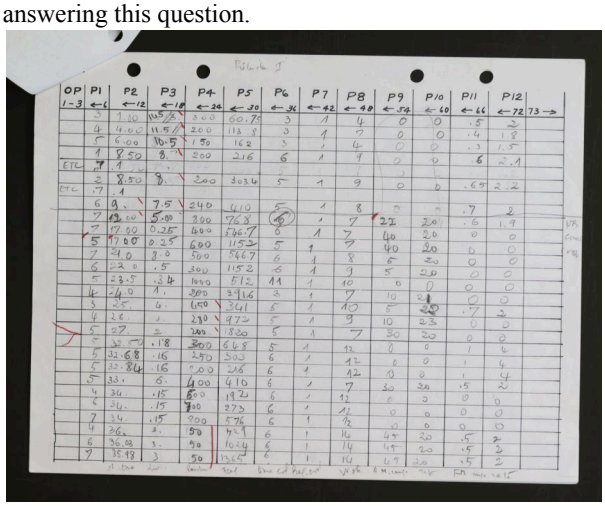


Figure 3. A card showing the content of the NOT section for *Prelude I*, R20_006, p.68.

We have retyped all the text containing the code and got fifty-one runs for *Little Boy* and ninety-four for *Mutations*. These were performed using our MUSIC V compiler, initially with mixed success. The failed runs have been used in debugging our source code restoration. We have also retyped much but not all of the score code for *Trumpet Study* and *Catalogue*. As part of our next steps, we will investigate and compare the content of the original magnetic tapes with our renditions of score code.

From this initial work, we may draw some interesting insights:

- Since we have the orchestra sections for some of the *Studies* from 1964/65 it should be possible to reconstruct and synthesise them in MUSIC V. This might enable us to get insights to the genesis of Risset's early composing style.
- *Study* examples, *Little Boy* and *Mutations I & II*, all use Pythagorean tuning extensively (Figures 1. - 4.). This has been concluded based on the observation of the score code, where NOT commands are given in Pythagorean tuning (A=432Hz). In the context of Risset's *microchirurgie sonore*[5] further perspectives open

[5] Sonic microsurgery: a term used by Risset to describe his handling of organised sound.

up: the relation of "large" major thirds to pure spectral thirds and to the inharmonic intervals used in bell-like sounds.

- Cross-referencing two versions of Mutations (I & II) might reveal interesting things on tuning. Our first findings suggest that he might have slightly modified the frequencies in Mutations II to match A=440, but kept the Pythagorean tuning (see Figure 4).
- A connection between *Microtone Studies* and *Little Boy* has been shown [27].

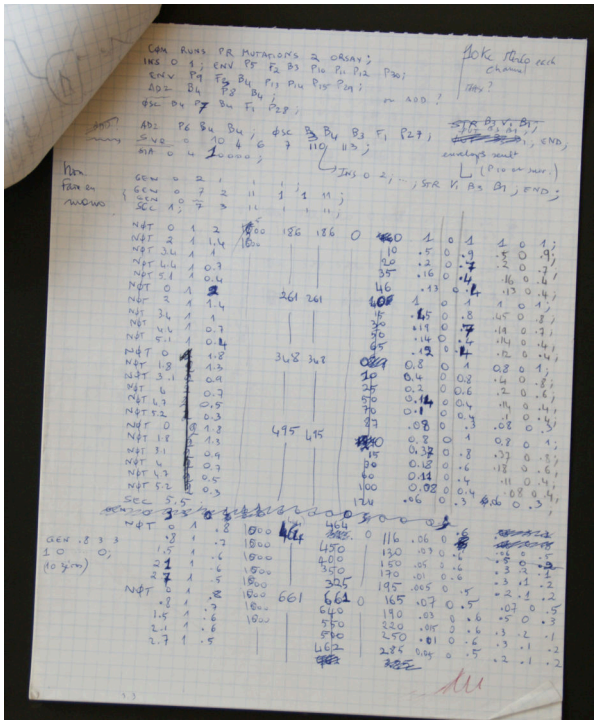


Figure 4. Risset's sketch for Mutations II, W20_005_1, p.39

5. CONCLUSIONS

This paper furnishes a description of methods and outcomes of an emergent field of study within ubiquitous music research: ubimus archaeology. We adopt the perspective outlined by Lazzarini and Keller [2] and tackle a two-pronged project. One branch deals with the computational issues involved in recovering the original version of the acoustic compiler MUSIC V to a multiplatform-deployable state that enables rendering musical works encoded during the 1960s. By making available the source code and a set of tools, the process of compiling MUSIC V code was streamlined. We hope this fosters future collaborations within the digital humanities. Another branch of the project entails recovering and analysing the earliest works done by Risset during his stay at Bell Labs. We chose *Little Boy*, *Mutations,* the *Introductory Catalogue of Computer Synthesized Sounds* and the *Computer Study on Trumpet Tones* as objects of study. Our analysis indicates a process of iterative explorations that includes both code-oriented work and annotations in English and French. Results obtained on one musical project often feed the development of other artworks. Thus, it is clear that a teleological method with explicit or fixed objectives was not adopted by Risset.

These results highlight new aspects of musicological endeavours and underline the diverse potential for application of ubimus frameworks. Ubimus projects have explored the expansion of technological sustainability through the adoption of repurposed hardware. Sometimes, this may imply inheriting the limitations and the biases of the hardware designs. As documented in Risset's annotations and in the multiple iterations of *Mutations* and *Little Boy*, the hardware constrains the compositional process, forcing the adoption of strategies to overcome some of the limitations. Complementarily, we can also infer that Risset's change of focus from the usage of alternative tunings to the explorations of new timbral possibilities and of perceptual phenomena was triggered by the creative possibilities provided by the MUSIC V environment. This raises a red flag on repurposing strategies. Stable technological infrastructures reduce electronic waste and are potentially good for the planet. But we may pay the price of reinforcing limited views on artistic activities and on music making.

Another issue illuminated by these results is replicability. The ubimus perspectives on replicability imply a delicate balance between scientific rigour and aesthetic diversity. By unearthing the original MUSIC V acoustic compiler we open the door to an exact replica of the code written in the sixties. Would this make sense in 2022? If it comes at the price of demanding the original hardware, then there are multiple arguments against it (see [5] for a detailed discussion of the caveats of this approach). While an exact replica may not be such a good idea, keeping the information in place and easily available makes a huge difference (as exemplified in Schottstaedt's version of the Fortran code). This procedure is aligned with Hunt and Thomas's [29] perspective on software archaeology. The source code is not enough. We need detailed comments, documentation and access to the history of design iterations. We also need to know the motivations for the implementation decisions. This approach is also applicable to the musical resources, as revealed by our analysis of Risset`s procedures.

ACKNOWLEDGEMENTS

We would like to acknowledge the support of the Fonds Jean-Claude Risset, PRISM, CNRS Marseille, in allowing us to access Risset's personal material, and especially to thank Dr. Vincent Tiffon, the head of the Fonds, for his kindness and availability. We would also like to thank Bill Schottstaedt for providing a machine-readable version of the Fortran code for MUSIC V. Damián Keller acknowledges support from CNPq [202559/2020-3]. Nemanja Radivojević acknowledges support from the Swiss National Science Foundation through the HKB BFH project "In Hommage from the Multitude".

REFERENCES

[1] D. Keller, V. Lazzarini, and M. Pimenta, *Ubiquitous Music*. Berlin and Heidelberg: Springer-Verlag, 2014.

[2] V. Lazzarini and D. Keller, "Towards a Ubimus Archaeology", In *Proceedings of the Ubiquitous Music Workshop*, https://doi.org/10.5281/zenodo.5553390, 2021.

[3] E. Hickmann, "Musikarchäologie." Begriffsbestimmung, Aufgabenfeld. In Laurenz Lütteken (ed.), MGG Online (2016–), 1997

[4] L. Candy, "Practice Based Research: A Guide", Technical Report, University of Technology, Sydney, 2006.

[5] N. Bernadini and A. Vidolin, "Sustainable live electroacoustic music," *eContact* 8(3), 2005.

[6] M. Johnson, Embodied knowing through art, In Michael Biggs and Henrik Karlsson, *The Routledge Companion to Research in the Arts*. London: Routledge, 2010, pp. 141-151

[7] R. Polfreman, R., D. Sheppard and L. Dearden, "Time to re-wire? Problems and strategies for the maintenance of live electronics," *Organised Sound* 11(3), 2006, pp. 229-242.

[8] P. Doornbusch, "Computer sound synthesis in 1951: The music of CSIRAC," *Computer Music Journal* 28(1), 2004, pp. 10-25.

[9] V. Lazzarini, "The Development of Computer Music Programming Systems", in *Journal of New Music Research*, 42 (1), 2013, pp. 92-110.

[10] M. Mathews, *The Technology of Computer Music*, MIT Press, 1970.

[11] J. C. Risset, *Introductory Catalogue of Computer Synthesized Sounds*. New Jersey: Bell Labs, 1969.

[12] B. Schottstaedt, "Pass1.f source code comments", https://github.com/vlazzarini/MUSICV/blob/master/src/pass1.f , 2008.

[13] R. Boulanger and V. Lazzarini, *The Audio Programming Book. Cambridge*, MA: MIT Press, 2010.

[14] V. Lazzarini, *MUSIC V code repository*, https://github.com/vlazzarini/MUSICV, 2021.

[15] F. Lévy, *Le compositeur, son oreille et ses machines à écrire. Déconstruire les grammatologies du musical pour mieux composer*, Vrin, 2013.

[16] O. Baudouin, *Pionniers de la musique numérique*, Delatour, 2012.

[17] D. Lorrain, "Inharmonique: analyse de la bande magnetique de l'oeuvre de Jean-Claude Risset", Rapport d'Ircam No. 26/80, 1980.

[18] A. Sousa Dias, *Jean-Claude Risset's Music V adaptations to MAxMSP*, 2009, https://github.com/asousadias/Risset_MaxPatches

[19] J. Svidzinski and V. Tiffon, *"*Jean-Claude Risset – Songes*", ANALYSES – Œuvres commentées du répertoire de l'Ircam,* 2021, https://brahms.ircam.fr/analyses/Songes/

[20] B. Schrader, *Introduction to Electronic Music*, Prentice Hall, 1982

[21] V. Tiffon, *Mémoire de Maîtrise*, Université de Tours, 1988.

[22] J.C. Risset, "Pitch Control and Pitch Paradoxes Demonstrated with Computer-Synthesized Sounds", The Journal of the Acoustical Society of America 46, 88, 1969.

[23] M. Mathews and J. Miller, *MUSIC IV Programmer's Manual*, Bell Labs, 1964.

[24] J. Beauchamp, "Review of Introductory Catalogue of Computer Synthesized Sounds", Perspectives of New Music Vol. 10, no. 1 (Spring/Summer - Autumn/Winter, 1971, pp. 348-350

[25] J. C. Risset, "My 1969 Sound Catalogue: Looking Back from 1992"*,* The Historical CD of Digital Sound Synthesis, CD Wergo 2033-2, 1995, pp.88-108.

[26] J. C. Risset, *Catalogue of Computer Synthesized Sounds,* accompanying record, Bell Telephone Laboratories, GRC-11874-A/B, 1969

[27] N. Radivojevic, "Zur Mikrotone Study von J.C.Risset", GMTh 2021: Tonsysteme und Stimmungen, Basel, 1st - 3rd Oct, 2021.

[28] V. Tiffon, Personal Exchange, 2020

[29] A. Hunt and D. Thomas, "Software archaeology" *IEEE Software*. March-April, 2002, pp.22-24.

Transformer and LSTM Models for Automatic Counterpoint Generation using Raw Audio

Lars Ødegaard Bentsen*
Department of Technology Systems
University of Oslo
lars.bentsen@its.uio.no

Riccardo Simionato*
Department of Musicology
University of Oslo
riccardo.simionato@imv.uio.no

Benedikte Wallace
Department of Informatics
University of Oslo
benediwa@ifi.uio.no

Michael Krzyzaniak
Department of Informatics
University of Oslo
michakrz@ifi.uio.no

ABSTRACT

A study investigating Transformer and LSTM models applied to raw audio for automatic generation of counterpoint was conducted. In particular, the models learned to generate missing voices from an input melody, using a collection of raw audio waveforms of various pieces of Bach's work, played on different instruments. The research demonstrated the efficacy and behaviour of the two deep learning (DL) architectures when applied to raw audio data, which are typically characterised by much longer sequences than symbolic music representations, such as MIDI. Currently, the LSTM model has been the quintessential DL model for sequence-based tasks, such as generative audio models, but the research conducted in this study shows that the Transformer model can achieve competitive results on a fairly complex raw audio task. The research therefore aims to spark further research and investigation into how Transformer models can be used for applications typically dominated by recurrent neural networks (RNN). In general, both models yielded excellent results and generated sequences with temporal patterns similar to the input targets for songs that were not present in the training data, as well as for a sample taken from a completely different dataset.

1. INTRODUCTION

Automatic counterpoint generation tasks aim to generate harmonically interdependent melodies added above or below a given melody. A melody can be considered as a group of notes played in sequence, one after the other, while a harmony is a group of simultaneous notes, played in the background and around the melody. Counterpoint is multiple concurrent melodies that follow a set of melodic rules, with respect to the sequential notes in each melody and all of the simultaneous notes in all of the melodies at each moment. Therefore, a model should generate a stylistically plausible counterpoint in order to create a single harmonic texture supporting a given melody. Such a problem can be considered as a branch of algorithmic musical composition, which has most commonly been used to support human creativity and develop tools to explore musical ideas, such as melodic or harmonic motifs.

Algorithmic musical composition have primarily been explored using symbolic music representations. In this paper we instead aim to explore the domain of algorithmic music composition by focusing on learning harmonic relations and dependencies from raw audio data. Recently, DL-based models have successfully been able to learn directly from raw audio [1]. As for most sequence-based applications, Convolutional (CNN) and Recurrent Neural Networks (RNN) have been predominant. However, a drawback of these methods is that they struggle to learn temporal dependencies across many time-steps, due to the limitations of CNNs' receptive fields and RNNs' internal state. WaveNet- and Long short-term memory (LSTM)-based models have been proposed to combat this impediment of CNNs and RNNs, respectively. WaveNet is a fully probabilistic and autoregressive model, which can be regarded as an extension of PixelCNNs [2], where the conditional probability distribution is modelled by a stack of convolutional layers. Dilated causal convolution is used, where dilation increases the model's receptive field by skipping input values with a certain step, and the causality ensures that future context is not taken into account when making a prediction. Several alterations of the WaveNet model have been explored for different audio applications, such as speech denoising [3], instrument conversion [4], vocoder [5], frequency estimation [6] and virtual modelling [7–9].

Despite the success of WaveNet, RNN-based models still seem the preferred choice for many audio applications, such as for virtual analog tasks [10, 11], where these models have even managed to meet tough real-time constraints [12, 13]. Furthermore, the accuracy of RNN-based architectures have often proved better than WaveNet-based models for black-box modelling of nonlinear audio systems, while requiring significantly less processing power to run [12]. A hybrid model, which used a combination of

* Equal contribution. Listing order is random.

J Copyright: © 2022 Bentsen et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

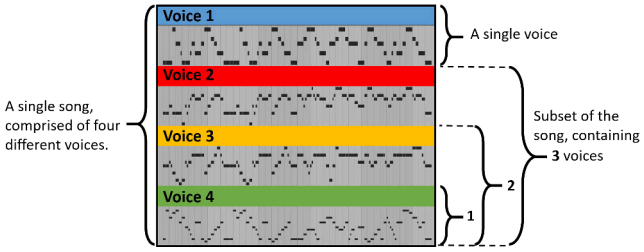


Figure 1: Example of MIDI representation of the four voices that comprise a single song.

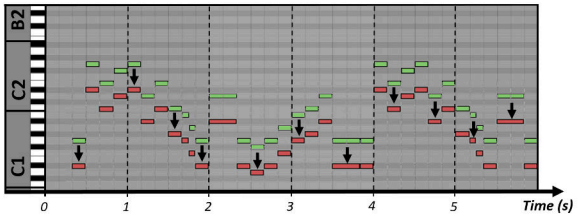


Figure 2: Example of pitch transposition for a single voice.

Input	Target
$[0,0,0,1]$	$[1,0,0,0]$
$[0,0,1,0]$	$[1,0,0,0]$
$[0,0,1,1]$	$[1,0,0,0]$
$[0,1,0,0]$	$[1,0,0,0]$
$[0,1,0,1]$	$[1,0,0,0]$
$[0,1,1,0]$	$[1,0,0,0]$
$[0,1,1,1]$	$[1,0,0,0]$
$[0,0,0,1]$	$[0,1,0,0]$
$[0,0,1,0]$	$[0,1,0,0]$
$[0,0,1,1]$	$[0,1,0,0]$
$[1,0,0,0]$	$[0,1,0,0]$
$[1,0,0,1]$	$[0,1,0,0]$
$[1,0,1,1]$	$[0,1,0,0]$
$[0,0,0,1]$	$[0,1,0,0]$
$(\cdots)$	$(\cdots)$
$[1,1,1,0]$	$[0,0,0,1]$

Table 1: Example showing the construction of the inputs and targets for a single song in binary encoding.

different DL architectures, was investigated in [14], where the model was divided into three parts; an adaptive front-end, latent-space and synthesis back-end. An important motivation behind the hybrid architecture, was to present a general-purpose DL framework for modelling audio effects.

After the Transformer was first introduced [15], it has outperformed many state-of-the-art architectures for a plethora of sequence-based applications, becoming increasingly popular after the release of the GPT-3 [16] and BERT [17] models. The success of these models for NLP tasks have later sparked research into the application of these models for various visual tasks, such as for image generation [18], image recognition [19–22], object detection [23], and segmentation [24, 25]. Transformer architectures have also started to make their way into the audio domain [26,27]. Different to RNNs, the Transformer relies on attention mechanisms to capture global context, instead of a recurrent unit with memory, making it potentially easier for the Transformer to capture temporal dependencies across much longer sequences than RNN-based models. This aspect is particularly important for music, as patterns can repeat or reappear after many time-steps, such as the chorus.

Considering the musical domain, Transformers have mainly focused on symbolic representations [26, 28], with only a few studies considering raw audio [29–31]. Li et al. [29] did not look at music, but introduced the Transformer TTS network for raw audio speech synthesis, which yielded unprecedented results, with mean opinion scores of 4.39, compared to 4.44 for real recordings. In the case of music, the analysis is typically more challenging than speech synthesis, as the latter usually considers a single speaker, while music signals are made up of multiple sources. Instruments like the piano are polyphonic, producing multiple pitches simultaneously, making the domain of music increasingly complex [32]. Child et al. [31] proposed the Sparse Transformer, which was able to model sequences of tens of thousands of time-steps, and showed impressive results generating raw audio classical music pieces of sequence lengths up to 65,536. Verma et al. [30] developed an auto-regressive and causal Transformer model for audio synthesis, using piano recordings from YouTube, containing both monophonic and polyphonic sounds. The inputs were waveform amplitudes, discretised in $0 - 255$, and the model aimed to predict the next value, one time-step ahead. They showed that the Transformer could be successfully applied to raw audio synthesis and outperform WaveNet-style models for certain tasks [30]. Aside from these studies, the literature is scarce with regards to the application of Transformers using raw audio data, and further research is therefore required to thoroughly explore the potential of such architectures, especially focusing on comparing against current state-of-the art.

The aim of this study was to generate a counterpoint for a given melody, using raw audio. With reference to Fig. 1, this meant generating a single missing voice, from a song containing any subset of the other three voices, shown here using symbolic (MIDI) representation for readability. The performance of the Transformer was investigated when applied to raw audio and compared against an LSTM model, which represents the current state-of-the-art. By studying a slightly more complex application than that in [30], this study aimed to further investigate the potency of the Transformer for use in raw-audio applications, which will be important to establish the architecture within the musical domain. Generally, the study investigated whether the Transformer could be successfully deployed and more extensively used for sequence modelling applications within music, which are different to the NLP domain.

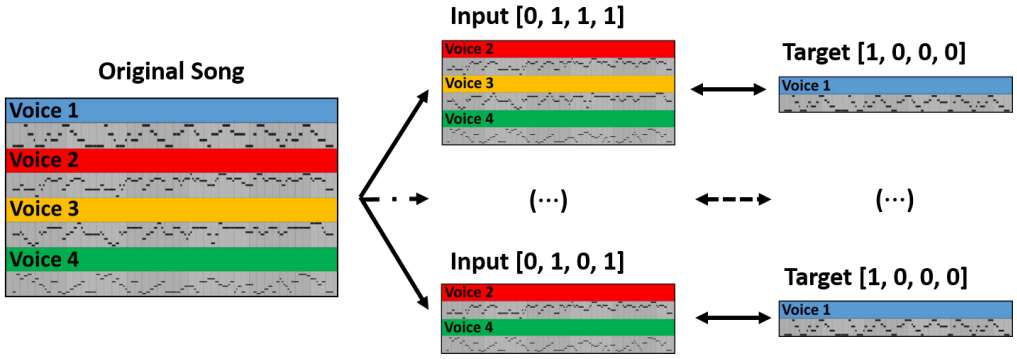


Figure 3: Illustration of the splitting of a song into inputs and targets, where the binary representation of inputs and targets are with reference to Table 1.

2. DATASET

The dataset used for this study was based on 5 different songs from some of Bach's work, each one composed of 4 voices in total. First, voices were transcribed in a MIDI format. To virtually increase the size of the dataset, the pitch was transposed up 5 semitones and down 6 semitones from the original, resulting in $(11 + 1)$ audio files for each song. The transposing of a single song, down 4 semitones, is illustrated in Fig. 2 and each of the 12 variations of a song therefore had different key roots. To produce the new representation of a song, all four voices (as depicted in Fig. 1) were transposed together, in the same direction and with the same magnitude. Instead of the 5 original songs, the new dataset now consisted of 60 different MIDI files.

The aim for this study was to propose a model that could predict a missing voice, given an input containing any subset of the remaining three voices in a song. Table 1 demonstrates the splitting of a single song into the corresponding inputs and targets using a binary representation, also illustrated in Fig. 3. Here, $[0, 0, 0, 1]$ indicates an audio file for a song, in which only voice 4 is present, $[0, 1, 1, 0]$, a file containing voices 2 and 3, and so on. Each possible combination was therefore extrapolated before rendering the raw audio files. This resulted in 15 combinations for each song, and 900 MIDI files in total. Files containing 4 voices were not considered, as the aim was to generate a missing voice from an input containing $1 - 3$ of the remaining voices, hence we obtain $16 - 1 = 15$ combinations for each song. Finally, the MIDI files were rendered to raw audio files using different instruments.

2.1 Data Preparation

The principal drawback of the Transformer, is that the complexity of its attention operations scales quadratically with sequence length. This impose significant computational and memory constraints, which typically limits the feasible sequence lengths that can be analysed. Because of the sequential architecture of an LSTM model, all relevant past information has to be stored in a single memory vector, resulting in these models struggling to capture temporal characteristics over a very large number of time-steps. LSTM networks could in theory be applied to arbitrarily long sequences, but because of the sequential architecture, there is typically an upper limit to the maximum feasible sequence length, as for the Transformer. Furthermore, the sequential architecture also means that recurrent models become slow for longer sequences. Since audio is sampled at tens of thousands of Hertz, the resulting raw audio sequences can be very long, even for short snippets of recording. For this particular study, each song was around 0.5 - 1.0 minutes long, which meant that, with a sampling rate of 44.1 kHz, each raw audio file would be represented by a sequence of around $1.3M - 2.6M$ values. Considerable effort was therefore spent in reducing input lengths to be able to use the Transformer and for the LSTM to be able to learn long-term temporal dependencies.

First, raw audio files were downsampled to 2940 Hz. By downsampling, the spectrum range was narrowed to a Nyquist frequency of 1470 Hz. Since notes present in the dataset were not higher than *DO6* (*C6*), i.e. 1046 Hz, all fundamental frequencies were still present in the new spectrum, while a large part of the overtones were discarded. This was decided as a trade-off between computational complexity, sequence lengths and frequency resolution.

For the task of counterpoint generation, spectral information, and in particular pitch information, was deemed the most important for the networks to learn. As a result, it was decided to use short-time Fourier transform (STFT), with a window length of 200 ms, to produce spectrograms from the raw audio files. Phase information was neglected and only magnitudes from the frequency spectra were taken.

The data was scaled in $(0, 1)$, using a standard Min-Max Scaler and split into test, train and validation sets. All files corresponding to a single, randomly chosen, song were used for testing and another for validation, while the files for the remaining three songs comprised the training set. Since there was no overlap between the test, train and validation sets, the aim was to improve regularisation by not allowing the models to overfit to the particular songs present in the training data.

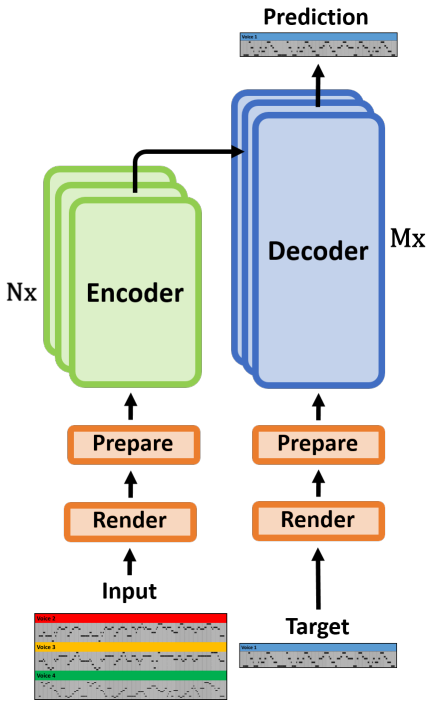


Figure 4: Proposed method for a generic encoder-decoder model, here taken as either an LSTM or Transformer model. *Render* refers to creating the raw audio files from MIDI representation. Even though the final output here are shown as MIDI, this was just for clarity, while the actual output was a raw audio file. Rendering, downsampling and STFT was done prior to training.

3. METHODS

3.1 Transformer

The Transformer model should take an input sequence, here a spectrogram, $x \in \mathbb{R}^{T \times d}$, and predict a missing melody, $y \in \mathbb{R}^{T \times d}$, where T is the number of time-steps and d the number of features (i.e. frequencies).

3.1.1 Encoder

The original Transformer architecture follows an encoder-decoder framework, as illustrated for a generic model in Fig. 4. The encoder is comprised of a multi-head attention mechanism and a traditional feed-forward neural network, as well as two residual connections and layer normalisations, as in the original formulation [15]. The input to the encoder is a sequence, which is added some positional encoding in order to capture temporal dependencies in the input. As in the original formulation of the Transformer, it was decided to use sine and cosine functions of different frequencies to represent the positional encoding.

The multi-head attention block in the encoder layer uses full self-attention, meaning that each attention operation would attend to the full input sequence. Multi-head attention, which computes multiple attention weights for updating each input, has also been found useful, as heads can learn to focus on different aspects of the input. Outputs from all heads are concatenated and passed through a learned linear transformation layer to produce the output.

The outputs from the multi-head attention block are added with a residual connection and applied layer normalisation, before being fed into a multilayer perceptron (MLP), typically with two hidden layers. The MLP is applied identically to all the inputs, individually. As for the multi-head attention, a residual connection and layer normalisation are applied to the outputs from the MLP, to produce the final outputs from the encoder layer. Finally, multiple encoding layers are stacked to introduce depth to the model, as shown in Fig. 4. The layers typically follow the same architecture, but each with different learnable weights, optimised through normal back propagation.

3.1.2 Decoder

The inputs to the decoder are the shifted outputs. First, the decoder employs an additional masked multi-head attention block, where the outputs are masked, so that the decoder only has access to $y_0, y_1, ..., y_{t-1}$ to make a prediction at time t, $\hat{y}_t$. The decoder then employs a multi-head attention block and MLP network in exactly the same manner as for the encoder. However, the inputs to compute the values and keys are now from the outputs of the final encoder layer, while the queries are from the outputs of the previous attention block in the decoding layer. This ensures that the model does not have access to outputs for future time-steps. As for the encoder, stacked decoding layers add depth to the model. For more details on the Transformer model and its implementation, the authors refer the reader to [15].

3.2 LSTM

Long short-term memory (LSTM) units were first introduced by Hochreiter et al. [33], in order to solve some of the shortcomings of vanilla RNNs when modelling long sequences. LSTMs partially solve the vanishing gradient problem [34] of RNNs, by employing a gated recurrent unit with skip-connections to allow gradients to flow across many time-steps. An LSTM unit consists of a cell, an input, output and forget gate. The cell stores values across multiple time-steps, acting as a memory, and the gates regulate the flow of information into and out of the cell. The forget gate controls what information in the cell state to forget, the input controls which new information will be encoded into the cell state, while the output controls what information encoded in the cell state is being outputted.

The LSTM model was also implemented in an encoder-decoder fashion, as shown in Fig. 4, where both the encoder and decoder could consist of multiple, stacked, LSTM layers. The encoder processes the input sequence and returns its final internal state and output. These values are then used as a sort of conditioning for the decoder, where the final state vector and output from the encoder set the first internal state of the decoder. In essence, the decoder therefore learns to predict the target at time-step t, given all past values of the target, $[y_0, y_1, ..., y_{t-1}]$, and conditioned on the input sequence. As for the Transformer, the decoder was trained to predict the target signal at the next time-step, given the previous outputs.

Model	NumParams	Architecture		LR	BS
Transformer	1,191,553	NumLayers:	3	0.001	16
		DimModel:	128		
		HiddenLayer:	256		
		NumHeads:	4		
		Dropout:	0.1		
LSTM	444,417	EncoderLayers:	1	0.001	16
		DecoderLayers:	1		
		EncoderUnits:	64		
		DecoderUnits:	64		
		OutputUnits:	256		
		Dropout:	0.1		

Table 2: Model configurations after hyperparamter tuning. LR and BS refers to learning rate and batch size.

Model	MSE	MAE
Transformer	$1.0404 \pm 0.003\text{e}{-5}$	$7.6733 \pm 0.2410\text{e}{-4}$
LSTM	$1.0388 \pm 0.004\text{e}{-5}$	$7.9989 \pm 0.5274\text{e}{-4}$

Table 3: Prediction Losses on the test dataset. Results are the mean taken from five different training iterations, along with the corresponding standard deviations.

3.3 Hyperparameter Tuning

To arrive at appropriate configurations for the models, a tuning process was conducted in Optuna, a framework for automated search of optimal hyperparameters [35]. The final architectures are summarised for both the LSTM and Transformer models in Table 2, where LR and BS refers to the learning rate and batch size, respectively. In addition to the parameters shown in the table, both the Stochastic Gradient Descent (SGD) and Adam optimisers were tested for, with the latter yielding the best results. The tuning was conducted in two stages. First, the models were trained to 500 epochs, before the search space was narrowed and models trained to 800 epochs. The final models were trained to 5000 epochs, when validation losses had properly converged for both models.

4. RESULTS & DISCUSSION

4.1 Prediction Losses

Both models were trained to minimise the Mean Squared Error (MSE), but also computing the Mean Absolute Error (MAE), given by the following equations:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_y)^2 \tag{1}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_y), \tag{2}$$

where n are the number of samples, y denotes the true labels (target) and $\hat{y}$ the predictions. Even though the two losses seem very similar, an important distinction is that the MSE metric will penalise large errors much more than the MAE metric. The authors also designed and tested various other loss functions, but this did not yield significant improvements for the final models and the MSE metric was therefore decided as the most appropriate loss function.

The final results for the LSTM and Transformer models when predicting on the test set are given in Table 3, as the mean and standard deviation from five different runs. Both models achieved excellent performance, with very small MSE and MAE values. The LSTM model yielded a slightly smaller MSE on the test set, while the Transformer performed better with regards to MAE. Nevertheless, due to the very small difference between the two models it was difficult to conclude on a better model out of the two. The fact that the Transformer achieved performance on-par with the LSTM model on a fairly complex raw-audio task, cements the hypothesis that Transformer-based architectures can be very competitive against the current state-of-the-art within sequence modelling. Having different models to choose from was thought desirable, as certain features of a particular architecture might be useful for the particular study or research problem.

4.2 Spectral Visualisation of the Results

Even though the MAE and MSE metrics were useful to evaluate the models, they are not very informative when trying to understand how good the models are at generating harmonies. The first two columns in Fig. 5 shows the generated spectrograms from the LSTM and Transformer models for two randomly selected test samples. The top row gives the true labels, i.e. what we wanted the models to predict, and the last two rows show the respective predictions. To listen to the different samples in Fig. 5, the raw audio files are also provided [1] . Looking at the first column, it was seen that both models generated samples that aligned closely with the target, as would be expected from the results in Table 3. The main melody was clearly captured by both models, as seen by the lower frequencies in the spectrograms. Interestingly, the models were also able to generate some of the, less pronounced, higher frequency harmonics. However, for the higher frequencies, the models tended to include more frequencies than were present in the target and it was more challenging to extract clear patterns for these. One reason for this drawback, might be that it was difficult for the models to learn the exact characteristics for frequencies with smaller amplitudes, as the models would not be penalised as heavily, with regards to MSE, for making wrong predictions for upper harmonics, compared to the main melody. Furthermore, the degree to which upper frequencies were present also varied significantly in the training samples, and it might be thought that the models slightly overfitted to the training data.

Moving to the second column of Fig. 5, the target seemed more challenging to predict than the previous sample, with much more frequencies present and generally lower amplitudes. Nevertheless, the models still performed well, capturing the main patterns present in the target, but seeming to include a few too many frequencies, some of which were not present in the target. The models also seemed slightly too decisive. In the target, it was seen that a number of frequencies close together were often present, while the Transformer and LSTM models generally seemed to only capture the main frequency in each group of frequencies.

[1] https://github.com/LarsBentsen/TransLSTM_RawAudio

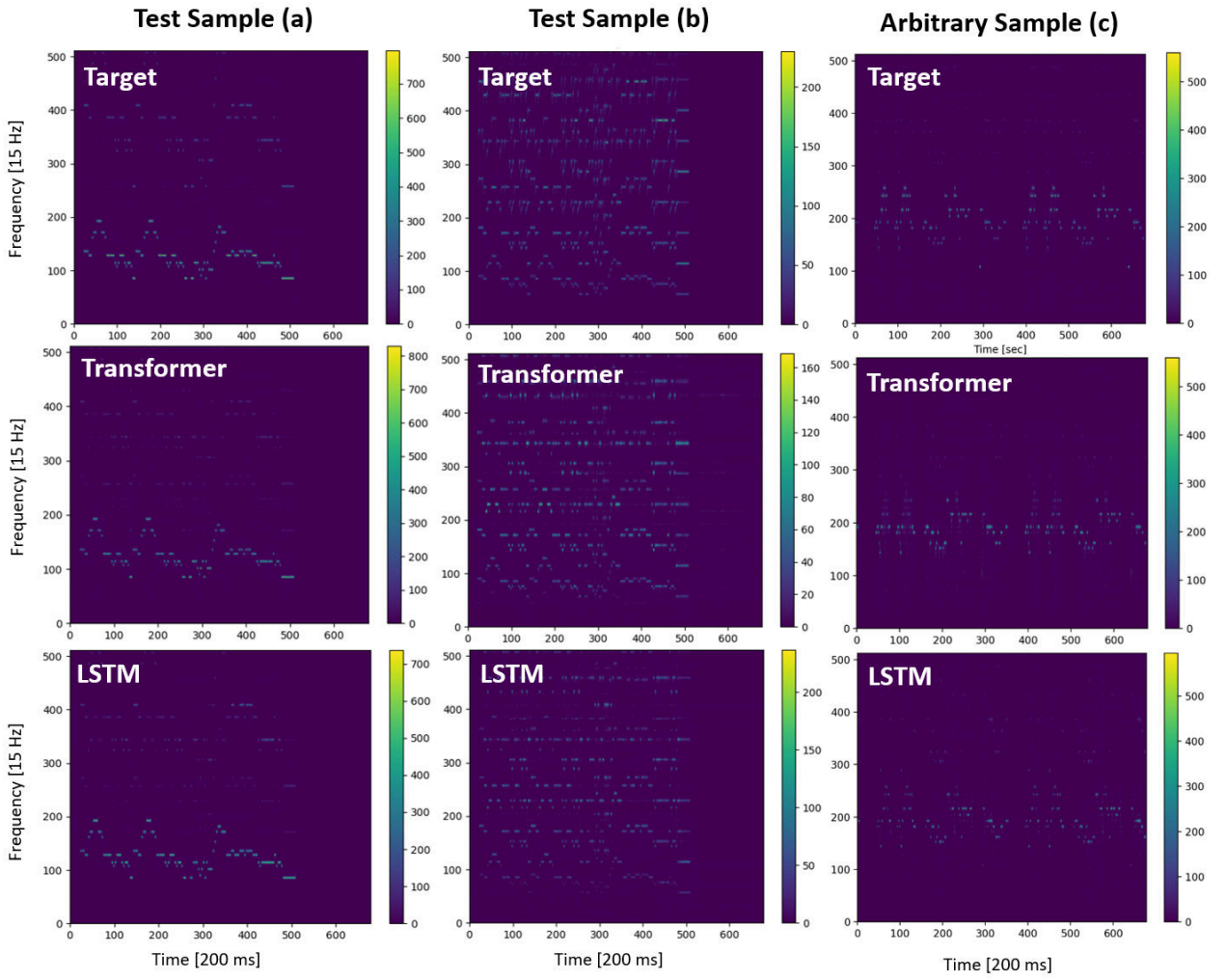


Figure 5: Spectral plots showing the predicted spectrograms and the corresponding targets (top row) for two randomly chosen samples from the test set and for one out-of-distribution sample (c), not taken from the original dataset. The plots are the raw output predictions from the Transformer and LSTM models, before performing inverse STFT and resampling, meaning that frequencies and times are given in 15 Hz and 200 ms resolutions, respectively.

Overall, it was seen through the spectrogram predictions and prediction losses that both models were well equipped to tackle the task at hand. Listening to the generated audio files, it was more challenging to distinguish between targets and predictions, than when considering spectrograms. Comparing the LSTM and Transformer models, the differences were minuscule, which proved the effectiveness of the Transformer for use on raw audio data.

The right hand column of Fig. 5, shows the Transformer and LSTM predictions for an arbitrary piece, which was not from the original dataset, but an extract taken from George Frideric Händel's Jubilate Deo (O be Jouyful in the Lord). It was seen that predictions from both models were slightly noisier than those for samples (a) and (b). A significant amount of higher frequency harmonics were predicted by the LSTM model, which were not present in the target. Nevertheless, both the LSTM and Transformer managed to predict the main melody reasonably well, indicating that the models were able to generalise to out-of-distribution samples, which further proved the prediction capabilities of both models.

For a given input, it is possible that a number of, completely different, melodies could follow the rules of counterpoint. Looking at the results in Fig. 5, it was seen that all predictions were very similar to the targets, even though ideal models should be able to generate a number of different melodies which fit the particular input. For this study, the models were not made autoregressive, but was conditioned on the previous, true, outputs to make the next predictions. Since the target spectrograms, in the top row of Fig. 5, were fed to the models with a look-ahead mask, it meant that the models were trained to output specific voices for a particular input. For future research, it will be particularly interesting to study ways in which the models could be made more creative, for which the authors of this paper think it would be important to make the models autoregressive and investigate ways in which the loss function could be altered. Since the MSE metric was used for

this study, the models would find a single melody for each input which minimised this loss. If instead, the loss function was founded on musical theory or more specific rules of counterpoint, it is thought that it might enable the models to generate more diverse sequences that can be different from the particular targets in the dataset.

Finally, the processing speed was estimated by running the models on a 2,3 GHz 8-Core Intel Core i9 processor. For a melody of 45.0 s, the Transformer computed all its predictions in 0.39 s, while the LSTM took 1.3 s. Furthermore, while the LSTM model took on average 3.5k epochs to converge, all Transformer models converged in < 50 epochs, significantly reducing training times.

For future work, the authors note a few additional areas of research thought particularly interesting. Since the complexity of the attention operations scale quadratically with sequence length, the model place some restrictions on the data that can be analysed. It would therefore be interesting to further investigate how the Transformer architecture could be adapted to reduce the complexity, such as the Longformer [36] and LogSparse Transformer [37], or other data preparation techniques which might be better equipped for extracting additional features in the raw audio samples. Further investigating architectures that facilitate the study of longer sequences is also thought to potentially improve the performance of Transformers over LSTMs. This is because the attention operation allows these models to attend directly to all previous time-steps for each computation, instead of storing all relevant information in a single memory array. The authors also believe that there is a significant potential with regards to the attention networks to improve the interpretability of the models. For instance, the Transformer relies on computing attention weights, which are analogous to how humans learn to focus on important aspects of a problem to arrive at a solution. By investigating a model's attention weights computed for a particular input, one could visualise which parts of the input the network learns to focus on. For music, this feature could be useful for a range of applications to better understand how the model learns and which parts of a sequence the model focuses on. Furthermore, the intuitive workings of the attention mechanisms are also thought interesting for allowing user inputs into a system to influence what to generate. However, this last point is not well studied and was merely thought as a potentially novel research direction for Transformer models in general.

5. CONCLUSION

In this study, the Transformer and LSTM models have been studied to generate counterpoint melodies from an input melody, using raw audio samples. Through analysis of the generated samples and prediction errors, it was shown that both models were well equipped for solving this problem. As a result, we argue that the Transformer model can be effectively deployed to musical applications, typically dominated by recurrent models. Due to the very long sequences when using raw audio samples, this study focused on downsampling and STFT to reduce the input lengths and create spectrograms. Furthermore, it was promising to see that both models could generalise well, producing appreciably accurate predictions for an out-of-distribution sample. For future work, it would be interesting to further investigate ways in which the Transformer and LSTM models could be altered to more effectively be used on raw audio data with very long sequences.

6. REFERENCES

[1] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6964–6968.

[2] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," *arXiv preprint arXiv:1606.05328*, 2016.

[3] D. Rethage, J. Pons, and X. Serra, "A wavenet for speech denoising," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5069–5073.

[4] N. Mor, L. Wolf, A. Polyak, and Y. Taigman, "A universal music translation network," *arXiv preprint arXiv:1805.07848*, 2018.

[5] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[6] P. Verma and R. W. Schafer, "Frequency estimation from waveforms using multi-layered neural networks." in *INTERSPEECH*, 2016, pp. 2165–2169.

[7] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time guitar amplifier emulation with deep learning," *Applied Sciences*, vol. 10, no. 3, p. 766, 2020.

[8] M. A. M. Ramírez and J. D. Reiss, "End-to-end equalization with convolutional neural networks," in *21st International Conference on Digital Audio Effects (DAFx-18)*, 2018.

[9] M. A. M. Ramirez and J. D. Reiss, "Modeling nonlinear audio effects with end-to-end deep neural networks," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 171–175.

[10] Z. Zhang, E. Olbrych, J. Bruchalski, T. J. McCormick, and D. L. Livingston, "A vacuum-tube guitar amplifier model using long/short-term memory networks," in *SoutheastCon 2018*. IEEE, 2018, pp. 1–5.

[11] T. Schmitz and J.-J. Embrechts, "Nonlinear real-time emulation of a tube amplifier with a long short time

memory neural-network," in *Audio Engineering Society Convention 144*. Audio Engineering Society, 2018.

[12] A. Wright, E.-P. Damskägg, V. Välimäki *et al.*, "Real-time black-box modelling with recurrent neural networks," in *22nd international conference on digital audio effects (DAFx-19)*, 2019.

[13] E.-P. Damskägg, L. Juvela, V. Välimäki *et al.*, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. Int. Sound and Music Computing Conf.(SMC-19), Malaga, Spain*, 2019, pp. 332–339.

[14] M. A. M. Ramírez, E. Benetos, and J. D. Reiss, "A general-purpose deep learning approach to model time-varying audio effects," *arXiv preprint arXiv:1905.06148*, 2019.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[16] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[18] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4055–4064.

[19] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *arXiv preprint arXiv:2103.00112*, 2021.

[20] X. Chu, Z. Tian, B. Zhang, X. Wang, X. Wei, H. Xia, and C. Shen, "Conditional positional encodings for vision transformers," *arXiv preprint arXiv:2102.10882*, 2021.

[21] Y. Tang, K. Han, C. Xu, A. Xiao, Y. Deng, C. Xu, and Y. Wang, "Augmented shortcuts for vision transformers," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[22] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," *arXiv preprint arXiv:2101.11986*, 2021.

[23] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[24] E. Xie, W. Wang, W. Wang, P. Sun, H. Xu, D. Liang, and P. Luo, "Segmenting transparent object in the wild with transformer," *arXiv preprint arXiv:2101.08461*, 2021.

[25] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr *et al.*, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6881–6890.

[26] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[27] P. Verma and J. Berger, "Audio transformers: Transformer architectures for large scale audio understanding. adieu convolutions," *arXiv preprint arXiv:2105.00335*, 2021.

[28] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[29] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6706–6713.

[30] P. Verma and C. Chafe, "A generative model for raw audio using transformer architectures," *arXiv preprint arXiv:2106.16036*, 2021.

[31] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[32] G. Peeters and G. Richard, "Deep learning for audio and music," 2021.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[35] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[36] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[37] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

Deep Embeddings for Robust User-Based Amateur Vocal Percussion Classification

Alejandro Delgado[1,2], Emir Demirel[1], Vinod Subramanian[1], Charalampos Saitis[1], and Mark Sandler[1]
[1] Queen Mary University of London [2] Luminary Roli Ltd.
a.delgadoluezas@qmul.ac.uk

ABSTRACT

Vocal Percussion Transcription (VPT) is concerned with the automatic detection and classification of vocal percussion sound events, allowing music creators and producers to sketch drum lines on the fly. Classifier algorithms in VPT systems learn best from small user-specific datasets, which usually restrict modelling to small input feature sets to avoid data overfitting. This study explores several deep supervised learning strategies to obtain informative feature sets for amateur vocal percussion classification. We evaluated the performance of these sets on regular vocal percussion classification tasks and compared them with several baseline approaches including feature selection methods and a speech recognition engine. These proposed learning models were supervised with several label sets containing information from four different levels of abstraction: instrument-level, syllable-level, phoneme-level, and boxeme-level. Results suggest that convolutional neural networks supervised with syllable-level annotations produced the most informative embeddings for classification, which can be used as input representations to fit classifiers with. Finally, we used back-propagation-based saliency maps to investigate the importance of different spectrogram regions for feature learning.

1. INTRODUCTION

Vocal Percussion Transcription (VPT) is a relatively old subfield in Music Information Retrieval (MIR) that is concerned with the detection and classification of vocal percussion sound events, sitting just between monophonic music transcription and speech recognition. The goal of VPT is to transcribe vocal percussion sound events into typical drum instrument classes, usually including kick drums, snare drums, and hi-hats. From here on, we will use the term *boxeme* (mix of "beatbox" and "phoneme") to refer to these vocal percussion sound events, as adopted in [1].

While VPT is a relatively specific field in MIR, its models and techniques are mostly shared across other disciplines in MIR and sound event detection. Some of these include musical instrument recognition [2], music transcription [3], query by vocal imitation [4], and anomalous sound detection [5]. Likewise, vocal percussion datasets are also used in areas like music cognition [6] and to study interpersonal differences and trends in vocal imitation styles [7, 8] among others.

Vocal percussion comes in two main modalities: beatbox and amateur vocal percussion. In *beatbox*, boxemes are produced using numerous parts of the vocal tract, including ones that are not used in normal speech [9]. This modality has its own universal set of techniques from which beatboxers base their own [10]. In contrast, *amateur vocal percussion* involves performers with little or no previous experience in vocal percussion, which usually includes most musicians and music producers. Due to this lack of training, boxemes are mostly speech-like and their articulation is much less consistent than in beatbox [9]. Also, as amateur performers do not follow vocal percussion techniques, they usually decide to use their own particular set of boxemes that differ from those of other performers [11].

The VPT process is usually composed by the onset detection and classification subtasks. *Onset detection* deals with the prediction of the exact moments in the audio waveform where boxemes start, whereas *classification* tries to assign each boxeme the correct associated drum instrument label (e.g. kick drums for /p/ boxemes). In this study, we focus exclusively on the classification process, leaving vocal percussion onset detection for future research.

Most recent works in amateur vocal percussion classification carry out the classification process in a user-based fashion, as this is known to improve classification performance compared to user-agnostic approaches [12–14]. In this way, users show the classifier their particular way of vocalising drum instruments (training set) so that the algorithm can recognise those boxemes in the future, usually within a beatbox-style improvisation (test set). Training sets are usually recorded either by reproducing a predictable beatbox-style phrase multiple times [15], which is called *fixed phrase* strategy, or by recording individual audio files containing same-class boxemes [11], which is called *isolated samples* strategy. Independently of the recording methodology, these user-specific training sets often contain less than a hundred boxemes. This *data bottleneck* limits the amount of input audio features that classifiers can take for modelling so as to minimise their risk of overfitting. In consequence, amateur vocal percussion classifiers are in need of naturally informative input feature sets to guarantee robustness in prediction accuracy for all participants, irrespective of their stylistic idiosyncrasies and vocal percussion skills. As the informative power of a

Copyright: © 2022 Alejandro Delgado et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

	AVP Dataset	LVT Dataset
Number of Participants	28	20
Number of Boxemes	4,873	841
Recording Strategy	Isolated Samples	Fixed Phrase
Instrument Labels	*kd, sd, hhc, hho*	*kd, sd, hhc*
Phoneme Labels	Yes	Yes

Table 1: Summary of datasets' contents (kd = kick drum, sd = snare drum, hhc = closed hi-hat, hho = opened hi-hat).

feature set depends largely on the task at hand, the exploration and evaluation of potentially informative feature sets would necessarily have to pass through boxeme analysis routines including heuristic feature selection and/or representation learning.

The present work explores the potential of several deep supervised learning approaches to generate informative feature sets for amateur vocal percussion classification. These feature learning strategies have been proven to be powerful feature extractors for high-dimensional data including sound events, music, and speech [16]. We supervise deep neural networks on four different types of label sets and take the values in their penultimate layer as the final feature sets to be evaluated. The four label sets that supervise the algorithms describe vocal percussion boxemes at different levels of abstraction, namely at instrument-level, syllable-level, phoneme-level, and boxeme-level. We assess the informative power of each of the learnt feature sets in terms of their classification accuracy and the stability of such metric to random train-validation splitting and initialisation routines. Finally, we carry out a complementary investigation of how relevant different regions in the spectrogram are for the models by applying saliency maps [17].

2. RELATED WORK

The use of VPT algorithms was perhaps first applied to beatboxing via [18] and [19]. These two studies feature three and two boxeme types respectively and both approach VPT by considering both the acoustic and rhythmic information contained in beatbox performances. In [20], Hazan proposed segmenting the boxemes in time prior to classification by applying an energy-based sound onset detector and explored K-Nearest Neighbours (KNN) and C4.5 tree algorithms for classification. This tree classifier was later revisited by Sinyor et al. [21], which explicitly included amateur vocal percussion boxemes in the evaluation dataset. Stowell et al. [22] conducted experiments on inter-class separability of feature vectors from beatbox sound events and discovered that a better classification performance in the real-time regime could be achieved by applying a 23-milliseconds delay to the start of the classification analysis frame from the actual onset. All these studies used heuristic feature extraction [23] and traditional machine learning methods [24] to train and evaluate boxeme classifiers.

Years later, novel approaches to VPT emerged inspired by recent advances in speech recognition, music information retrieval, and sound event classification techniques. In [25], Picart et al. explored audio pitch-tracking as a complement to the sound onset detection and classification engines, using a Hidden Markov Model (HMM) for transcription. This study also featured vocal imitations of pitched sounds from several musical instruments, where pitch-tracking algorithms proved the most useful. Ramires recorded in [15] the Live Vocalised Transcription (LVT) dataset, which was the first publicly available amateur vocal percussion dataset, and later developed the homonymus LVT system based on KNN vocal percussion classifiers [13]. In this study, we only use the acronym "LVT" when referring to the dataset. Delgado and colleagues [11] recorded another large publicly available amateur vocal percussion set, the Amateur Vocal Percussion (AVP) dataset, and carried out an evaluation of several onset detection algorithms. The same authors later explored data augmentation and deep learning techniques for user-based boxeme classification [14], where Convolutional Neural Network (CNN) models [26] achieved the best accuracy results. Finally, Evain et al. [1] adapted a popular HMM-based tool for automatic speech recognition [27] to VPT. The model was trained on a corpus of eighty different boxemes recorded individually by two beatboxers and yielded the best results using 22 Mel Frequency Cepstral Coefficients (MFCC) and their derivatives as the input features.

In the present study, we build on the works above and address several shortcomings of these, especially when it comes to final real-world implementations. These limitations include (i) the little availability of publicly available vocal percussion datasets, which impacts reproducibility, (ii) the low amount of participants and small size of the available datasets, which impacts the soundness and generalisability of results, (iii) the lack of beatbox-like improvisatory performances in some studies, which impacts the validity of results' extrapolation to real-world scenarios, and (iv) the lack of publicly available code, which impacts transparency. We tackle these issues respectively by (i) joining two publicly available datasets for amateur vocal percussion (AVP [11] and LVT [15]) and adapting the mix to VPT evaluation routines, (ii) using audio data augmentation techniques to improve external generalisability, (iii) testing final algorithms on freestyle improvisations to better assess their real-world capabilities, and (iv) publishing our code in an open-source repository [1] .

3. METHODOLOGY

In this section, we provide an in-detail account of the main data sources, algorithms, and routines used throughout our study. In section 3.1, we talk about the two datasets that we used (AVP and LVT), how we joined them and expanded their annotations so as to include phonetic information, and how we built input representations and carried out the data augmentation process. In section 3.2, we describe the architecture of the embedding learning model and the seven types of label sets that we used for its supervision. In sec-

[1] *https://github.com/alejandrodl/vocal-percussion-transcription*

Onset Phonemes	Coda Phonemes
/t/ and /!/	/ɑ/, /æ/, /ɐ/, and /ʌ/
/ts/ and /s/	/e/, /œ/, and /ə/
/tʃ/, /tɕ/, /dʒ/, and /tʒ/	/i/, /y/, and /ɪ/
/kx/, /k/, and /kʃ/	/o/ and /ʊ/
/p/ and /ʔʕ/	/u/ and /ɯ/

Table 2: Phoneme groupings for the reduced sets.

tion 3.3, we present the three baseline methods whose performances were compared to those of the embedding learning model supervised on different label sets and in section 3.4 we show how both the training and the evaluation processes were carried out.

3.1 Data and Pre-Processing

We used two publicly available vocal percussion datasets throughout the study: the AVP dataset [11] and the LVT dataset [15]. We contrast some of these datasets' characteristics in Table 1. The AVP dataset contains a total of 9778 boxemes (4873 and 4905 from the personal and fixed subsets respectively) recorded by 28 participants and with four annotated labels: kick drum, snare drum, closed hi-hat, and opened hi-hat. Its training dataset was recorded using the isolated samples strategy. To train our acoustic models, we exclusively used the personal subset (participants vocalising boxemes of their choice), although we also used the fixed subset (participants vocalising common boxemes) to train the sequential module of the baseline speech recognition model (see section 3.3). The LVT dataset contains a total of 841 boxemes recorded by 20 participants with three annotated labels: kick drum, snare drum, and closed hi-hat. Here, the training dataset was recorded using the fixed phrase strategy. Also, we exclusively use its third subset, as the recordings' quality and background noise level are similar to those from the AVP dataset.

We manually expanded the annotations (onsets and instrument labels) of both datasets so as to include the syllabic representation of boxemes. The syllables were composed of a first *onset phoneme*, usually plosive or fricative, and a second *coda phoeneme*, usually a vowel, a breath sound, or silence (no coda phoneme). These phonemes were annotated following notation conventions from the International Phonetic Alphabet (IPA). Apart from the *original* phoneme set, we also elaborated a *reduced* phoneme set in which several similar-sounding phonemes were put together to form single classes. For this reduced version of phoneme annotations, onset and coda phonemes were grouped as shown in Table 2. We also make final AVP-LVT dataset with expanded annotations publicly available [2] .

Joining the AVP and LVT datasets, we had a total of 5714 boxemes. As this amount of data was relatively modest for deep embedding modelling, we applied *waveform data augmentation* to boxemes, specifically random *pitch-shifting* (semitone range = [-1.5,+1.5]) and *time-stretching* (stretch factor range = [0.8,1.2]), one after the other in random order. This kind of data augmentation is standard in audio signal processing [28] and it has been proven to improve the accuracy of vocal percussion classification algorithms [14]. We applied ten iterations of random data augmentation in this manner, ending up with a total of 62854 boxemes in the final dataset.

[2] *https://zenodo.org/record/5578744#.Yfpu9PXP30o*

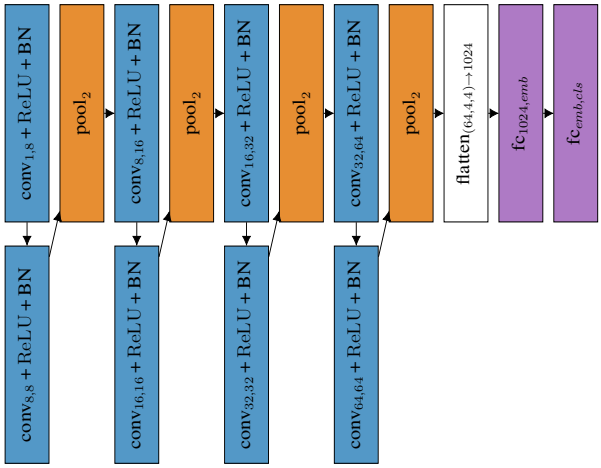


Figure 1: Diagram of the CNN embedding learning model. The variable emb refers to the number of final embeddings to be extracted by removing the last layer and cls refers to the number of classes with which the model is pretrained.

As input to neural networks, we built Mel spectrogram representations from each boxeme using 64 Mel frequency bands and a hop size of 12 milliseconds. We used 48 time steps ($\sim$ 0.56 seconds) so that the final boxeme spectrograms had a final dimension of 64x48 and we explored frame sizes of 23, 46, and 93 milliseconds, ultimately reporting the one that brought the best results in Section 3.1. We post-processed spectrograms with a logarithmic transform ($log(spectogram + 0.0001)$) and normalised them to a [0,1] range.

3.2 Supervised Embedding Learning

We used the penultimate layers of several CNN classifier models as the final embeddings to perform evaluation on. The architecture of these CNN models is illustrated in Figure 1. They all had four convolutional blocks with 8, 16, 32, and 64 filters respectively and two fully-connected (FC) linear layers, one connecting the flattened feature maps to the embedding space and another one connecting the embedding space to the labels. Each convolutional block had two convolutional layers with kernels of size 3x3 and stride 1x1, each one followed by a batch normalisation module and a ReLU activation gate. A final max-pooling operator with kernel size 2x2 is applied at the end of each convolutional block so as to progressively downsample the feature maps. The design of the network's architecture and its training routines (see section 3.4) was inspired by best practises in CNN-based research [29, 30].

We explored seven different supervision strategies to train and validate the above mentioned CNN classifiers. These strategies use different types of labels that describe the same input data at different levels of abstraction.

3.2.1 Instrument-Level Annotations

We used two types of drum instrument annotations: the ones relative to the *original* set (kick drum, snare drum, closed hi-hat, and opened hi-hat) and the ones relative to a *reduced* version of it (drum and hi-hat). These constituted the first and the second supervision strategies.

3.2.2 Syllable-Level Annotations

We also used syllable labels, put together by joining the onset and coda phonemes in the *original* and the *reduced* phoneme sets (see table 2). These constituted the third and the fourth supervision strategies.

3.2.3 Phoneme-Level Annotations

We used individual phoneme labels, which also came from the *original* and the *reduced* phoneme sets. These constituted the fifth and the sixth supervision strategies. It is worth noting that, while the two phoneme-level sets contained the same information as the two syllable-level ones, here the CNN classifier predicts onset and coda phoneme labels separately in a multi-task way, managing two different validation losses and accuracies.

3.2.4 Boxeme-Level Annotations

Finally, we used boxeme labels to describe boxemes at the lowest level of abstraction. These boxeme classes were integrated by sounds that had different associated syllables and also pertained to different participants. This constitutes the seventh supervision strategy.

3.3 Baseline Algorithms

We compare the performance of learnt embeddings with two baseline heuristic feature sets and a speech recognition model. We used the Essentia toolbox [31] to calculate audio descriptors and the Kaldi library [27] to implement the speech recognition model.

3.3.1 Timbre Feature Set

The first baseline model is the *timbre feature set*, which is made of Mel Frequency Cepstral Coefficients (MFCC) and envelope features. The MFCCs are computed using the same frame-wise analysis parameters as for the spectrogram representations. For the 32-dimensional feature vector (see Section 3.4), we take the mean of the first 14 MFCCs, the mean of their first derivative, and four envelope descriptors: the derivative after the maximum amplitude, the maximum derivative before the maximum amplitude, the flatness coefficient, and the temporal centroid to total length ratio. For the 16-dimensional feature vector, we take the mean of the first 12 MFCCs along with the four envelope descriptors.

3.3.2 Feature Selection

For this method, we extracted 258 heuristic features from the spectrum and the envelope of boxemes and implemented seven importance-based *feature selection* algorithms whose base algorithms were supervised using the same seven supervision strategies described in Section 3.2. Here, instead of learning embeddings from spectrogram representations, we extracted the set of heuristic features, derived feature importances from a random forest base algorithm through a 10-way feature permutation process [32], and selected the most informative features to build the final set. In the case of phoneme-level supervision strategies, the final selected features were drawn from the intersection of the two independent feature importances arrays, gathering the features that were considered most important to classify both onset and coda phonemes. We only reported the best result from these seven approaches in Section 4.

3.3.3 GMM-HMM Speech Recogniser

For the last baseline method, we tackled vocal percussion classification by means of *speech recognition*, an approach proposed by Evain et al. in [1] that was originally applied to beatbox classification with satisfactory results and that we bring to amateur vocal percussion classification in order to provide context for our methods. Here, each instrument type to detect is considered as a word to be recognized. First, an *acoustic model* was trained to learn a relationship between acoustic speech features and phonemes. The mapping between instrument types and their corresponding phonemes was established through a pronunciation dictionary, converting phoneme posterior probabilities into word probabilities. We exploited both instrument- and phoneme-level annotations to construct this pronunciation dictionary. Word probabilities were then smoothed using a language model so as to obtain grammatically sensible transcriptions. The *language model* was 5-gram trained on the transcriptions from training data, and the recogniser was trained via the Kaldi GMM-HMM recipe using the code in *https://github.com/emirdemirel/ALTA* [33]. This was a triphone model trained with speaker adaptive features [34]. Hyperparameters were determined empirically and they are available in the paper's repository.

3.4 Training and Evaluation

A schematic diagram of the training and evaluation processes is provided in figure 2.

We chose four participants from the AVP dataset and four from the LVT dataset to compose the evaluation set. Two women and two men per dataset were selected for evaluation on the basis of acceptable pronunciation and overall representativeness of the dataset. In the end, we had a total of 8 participants in the evaluation set and 40 participants in the train-validation set. We provide the distribution of the instrument, syllable, onset phoneme, and coda phoneme labels in the project's code repository.

We trained our seven CNN models (see sections 3.2.1, 3.2.2, 3.2.3, and 3.2.4) using the train-validation dataset. We set up a 5-fold cross-validation routine in which models were trained and validated for 5 iterations per fold, each with different initialisation parameters. Therefore, we end up with 25 deep embedding models per supervision strategy that encapsulate two main sources of training arbitrariness: the content of train-validation splits and random weight initialisation. We compute the mean and the 95% confidence intervals of the subsequent 25 evalua-

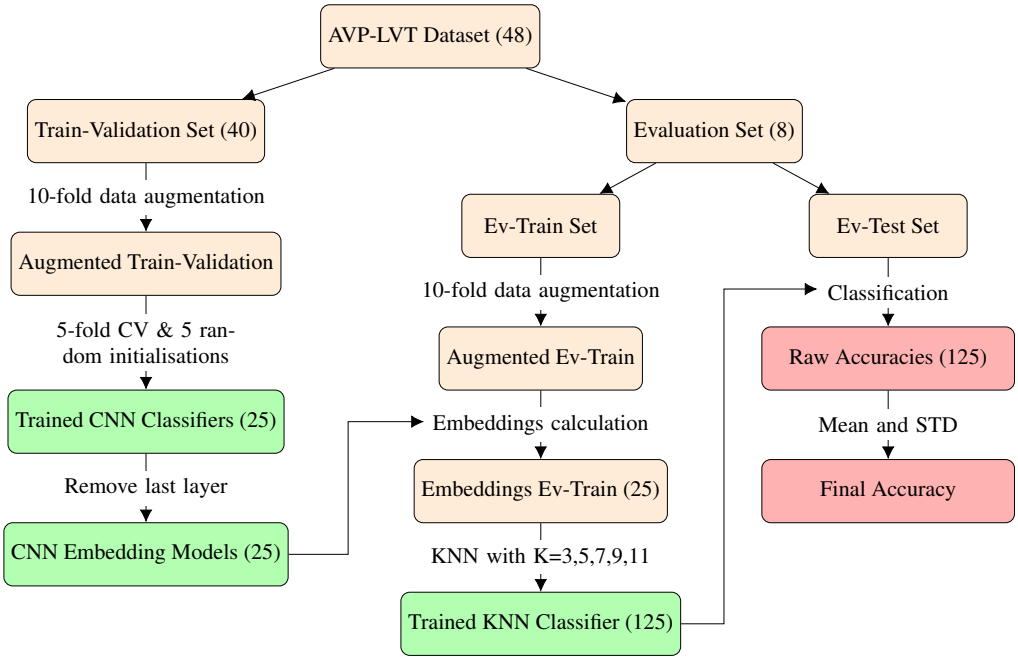


Figure 2: Diagram of the training-evaluation process. Background colour code: orange = dataset-related, green = model-related, red = results-related.

tion performances when reporting final results for a given supervision strategy. The same train-validation arrangement applies to the seven baseline feature selection methods. We trained the models using an Adam optimisation algorithm [35], early stopping if validation loss has not decreased after 10 epochs, and a further regularisation routine that downscales the learning rate if validation loss has not decreased after 5 epochs. In the case of phoneme-level supervision, we explored two settings of loss weights to compute the joint loss value after each training batch. The onset-coda weight percentages for those two settings were 50-50% and 60-40%.

As explained in section 3.1, each of the 8 participants in the evaluation set have their own train and test subsets. From here on, we refer to these as the *ev-train* and the *ev-test* sets respectively to improve readability. During evaluation, a KNN algorithm was trained on the ev-train set of each participant and evaluated on the ev-test, taking the learnt CNN embeddings and the baseline feature sets as input representations. As these embeddings and feature sets are expected to have different information for the KNN algorithm to rely on, they are also expected to make the classifier perform better or worse given the underlying suitability of these feature sets for classification. As KNN algorithms are purely distance-based and non-parametric, our assumption is that a high classification accuracy using them is more likely to translate into a high accuracy using other types of machine learning algorithms. In Section 4, we had the opportunity of briefly testing this hypothesis by replacing the KNN classifier with three other popular machine learning classifiers whose results are commented but not reported.

The evaluation procedure above is applied to all proposed and baseline methods except for the GMM-HMM-based speech recognition model. This model does not extract embeddings, but rather predicts the ev-test labels directly ignoring the ev-train subset (user-agnostic). Also, to alleviate the disparity in the amount of ev-train data in the AVP and the LVT datasets (1,000 vs. 220 augmented data samples approx.), we extract two different amounts of embeddings and selected features: 32 to carry out evaluation on AVP participants and 16 to do it on LVT ones. This means that for each supervision method we end up having 25 feature sets of size 32 to evaluate on AVP data and other 25 of size 16 to do it on LVT data.

We report final results in two modalities: *participant-wise*, where accuracy scores of single test participants are calculated and averaged, and *boxeme-wise*, where accuracy scores are calculated for all evaluation boxemes in a participant-agnostic way.

Finally, we used *saliency maps* [17] to highlight sections of the input that are important for model prediction. These maps can be computed given an input x and the penultimate layer output of a deep learning model A as: $\frac{dA_i}{dx}$, where i is the label with respect to which the saliency map is computed. In our case, A is the CNN model conditioned on the original instrument labels. We aggregated the saliency map by taking the absolute value and thresholding it, so that the top 10% of the map is set to one and the rest is set to zero. Then we averaged the saliency map over the 25 models.

4. RESULTS

4.1 Classification

Final evaluation accuracies for all methods are gathered in Table 3. Best results were obtained using a frame size of

	Baseline (3.3.{1-2-3})			Instrument (3.2.1)		Syllable (3.2.2)		Phoneme (3.2.3)		Boxeme (3.2.4)
	Timbre	Selection	HMM*	Original	Reduced	Original	Reduced	Original	Reduced	Original
Part-wise	.840	.827±.030	.725	.812±.037	.779±.034	**.899±.025**	.883±.030	.876±.028	.874±.030	.861±.030
Box-wise	.835	.795±.011	.734	.774±.038	.738±.033	**.874±.029**	.852±.031	.840±.029	.838±.032	.832±.031

* *User-agnostic model*

Table 3: Final evaluation accuracies from generated feature sets. Results are given participant-wise and boxeme-wise, and best performances for both modalities are highlighted in bold font. For feature selection, only the best performance is reported (reduced syllable-level). For embedding learning (25 models), the mean performances and their 95% confidence intervals are reported.

46 ms and, in the case of phoneme-level classification, loss weights of 0.6 and 0.4 for onset and coda phonemes. Also, the best-performing feature selection routine was the one using the original syllable label set.

We can observe in the table that all supervised embedding models except for those supervised with instrument-level classes are consistently superior to baseline approaches, including feature selection approaches. This observation lies in accordance with previous literature on the usefulness of deep learning models as feature extractors for speech utterances [16]. It also highlights the unsuitability of instrument-level classes for embedding learning supervision, which was somewhat expected given that participants have different ways of vocalising drum instruments. Thus, label sets of such (high) level of abstraction are undesirable for embedding learning in our case.

We see that the best performance for both participant-wise and boxeme-wise evaluation metrics is achieved by supervised embedding learning models that used the original syllable-level label set. This difference is notable not only for the high mean accuracy score but also for its 95% confidence intervals, which is the lowest for both participant-wise and boxeme-wise metrics. This means that the informative power of the resulting embeddings, apart from being the most prominent, is also the most robust to the two sources of training arbitrariness that we studied here (see Section 3.4). We carried out experiments using other different classifiers than KNN, namely logistic regression, random forest, and extreme gradient boosted trees. There, we observed that the original syllable-level method still outperforms the rest of approaches, which further reinforces its suitability for classification. All methods performed similarly on these extra algorithms, both in terms of absolute and relative performances.

Models supervised using original and reduced phoneme-level classes also yielded similar scores to those of the ones with syllable-level supervision, although still lower and generally less robust to training arbitrariness, possibly due to the extra complexity of the multi-task learning approach. The same applied to boxeme-level supervision, which performed slightly worse than phoneme-level supervision, possibly indicating that very low levels of supervision abstraction are relatively counterproductive for vocal percussion classification engines.

Another reason that could explain this lower performance for boxeme-level supervision could be its large amount of output labels (∼150 boxeme types), which complicates validation. In order to tackle this issue, we built and trained an alternative siamese network model [36] with the same architecture as the CNN except for the last fully connected layer, which was removed. This network uses metric learning directly on embeddings to discriminate between same-class boxeme pairs and different-class ones, therefore reducing our large label vector to a "same-different" binary auxiliary vector. In the end, its final accuracy was found to be moderately lower than the one relative to the CNN classifier, so we kept the latter's result.

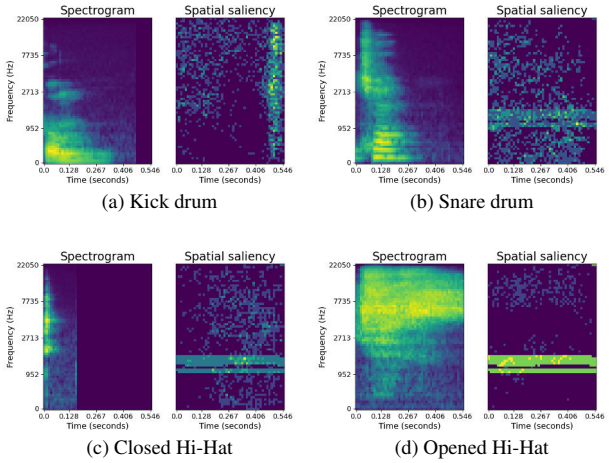
We also notice that the generic (user-agnostic) HMM-based speech recognition model performs worse than any other user-based method. This result evidences the importance of taking user idiosyncrasies into account in amateur vocal percussion classification, making user-based strategies preferable to generic user-agnostic ones. Finally, we observe that accuracies derived using the timbre feature set are higher than all the ones pertaining to baseline feature selection algorithms. This result could indicate an excessive information redundancy of selected features compared to that of the timbre set, which is a more internally cohesive feature set.

A clear limitation of the syllable-level embeddings as inputs to amateur vocal percussion classifiers is that these were learnt using samples recorded with electronic devices in a small room with little background noise, which emulates the typical recording setting of amateur music producers. This is likely to work similarly well in recording studios, where audio quality is higher, but may very well lose a significant part of its accuracy when faced with low-quality recordings or contexts with too much noise. The timbre feature set could be of great help for these situations, as its features are context-agnostic and therefore adapt well to challenging recording scenarios.

4.2 Saliency Maps

Four representative examples [3] of these maps are shown in Figure 3. In general, we found that models tended to focus more on frequencies between $1000Hz$ and $2000Hz$ for boxemes associated with snare drum, closed hi-hat, and opened hi-hat. This region usually coincides with a high-energy one for these boxemes, which often share phonetic representations. It also might indicate a useful thresholding point for the network to tell the boxemes apart. Hence,

[3] See project's repository for more examples of saliency maps.



(a) Kick drum (b) Snare drum

(c) Closed Hi-Hat (d) Opened Hi-Hat

Figure 3: Log mel spectrogram (left) and corresponding saliency map (right) of four boxemes of different instrument class.

this could be implying that a higher frequency resolution in this region could potentially improve the accuracy of future amateur vocal percussion classifiers.

The network also appears to attend to silences and regions of lower spectral energy in the case of the kick drum and closed hi-hat. This could mean the absence of energy, especially in the high end of the spectrum, might also be a key feature for the models to differentiate the kick drum and closed hi-hat from the rest of the instruments. The high attention density in silences specifically could also be implicitly suggesting that the duration of boxemes is a key factor to distinguish the kick drum and closed hi-hat from the snare drum and opened hi-hat, as the boxemes associated with these last two instruments tend to be longer.

5. CONCLUSION

We have explored the capabilities of supervised embedding models to generate informative feature sets that can achieve amateur vocal percussion classification in a fast and reliable way. We used seven supervision strategies whose label sets describe vocal percussion sound events (boxemes) at different levels of abstraction and we found that CNN classifiers supervised using syllable-level classes not only produced the best mean accuracy results but also were the most robust to sources of training randomness like weights initialisation. As discussed in the results section, we highly encourage the use of these embedding models as feature extractors for amateur vocal percussion classifiers in recording settings with low and moderate background noise. We have made both the AVP-LVT dataset and the code for this study publicly available.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068.

6. REFERENCES

[1] S. Evain, B. Lecouteux, D. Schwab, A. Contesse, A. Pinchaud, and N. H. Bernardoni, "Human beatbox sound recognition using an automatic speech recognition toolkit," *Biomedical Signal Processing and Control*, vol. 67, p. 102468, 2021.

[2] A. Solanki and S. Pandey, "Music instrument recognition using deep convolutional neural networks," *International Journal of Information Technology*, pp. 1–10, 2019.

[3] M. A. Román, A. Pertusa, and J. Calvo-Zaragoza, "An end-to-end framework for audio-to-score music transcription on monophonic excerpts." in *ISMIR*, 2018, pp. 34–41.

[4] A. Mehrabi, K. Choi, S. Dixon, and M. Sandler, "Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 356–360.

[5] E. C. Nunes, "Anomalous sound detection with machine learning: A systematic review," *arXiv preprint arXiv:2102.07820*, 2021.

[6] A. Mehrabi, S. Dixon, and M. Sandler, "Vocal imitation of percussion sounds: On the perceptual similarity between imitations and imitated sounds," *Plos one*, vol. 14, no. 7, p. e0219955, 2019.

[7] A. Delgado Luezas, C. Saitis, M. Sandler *et al.*, "Spectral and temporal timbral cues of vocal imitations of drum sounds." International Conference on Timbre, 2020.

[8] A. Delgado, C. Saitis, and M. Sandler, "Phoneme mappings for online vocal percussion transcription," in *Audio Engineering Society Convention 151*. Audio Engineering Society, 2021.

[9] N. Patil, T. Greer, R. Blaylock, and S. S. Narayanan, "Comparison of basic beatboxing articulations between expert and novice artists using real-time magnetic resonance imaging." in *Interspeech*, 2017, pp. 2277–2281.

[10] D. Stowell and M. D. Plumbley, "Characteristics of the beatboxing vocal style," *Dept. of Electronic Engineering, Queen Mary, University of London, Technical Report, Centre for Digital Music C4DMTR-08-01*, 2008.

[11] A. Delgado, S. McDonald, N. Xu, and M. Sandler, "A new dataset for amateur vocal percussion analysis," in *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, 2019, pp. 17–23.

[12] K. Hipke, M. Toomim, R. Fiebrink, and J. Fogarty, "Beatbox: End-user interactive definition and training

of recognizers for percussive vocalizations," in *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, 2014, pp. 121–124.

[13] A. Ramires, R. Penha, and M. E. Davies, "User specific adaptation in automatic transcription of vocalised percussion," *arXiv preprint arXiv:1811.02406*, 2018.

[14] A. Delgado, S. McDonald, N. Xu, C. Saitis, and M. Sandler, "Learning models for query by vocal percussion: A comparative study," in *Proceedings of the International Computer Music Conference*, 2021.

[15] A. F. S. Ramires, "Automatic transcription of vocalized percussion," 2017.

[16] S. Latif, R. Rana, S. Khalifa, R. Jurdak, J. Qadir, and B. W. Schuller, "Deep representation learning in speech processing: Challenges, recent advances, and future trends," *arXiv preprint arXiv:2001.00378*, 2020.

[17] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps." [Online]. Available: http://arxiv.org/abs/1312.6034

[18] A. Kapur, M. Benning, and G. Tzanetakis, "Query-by-beat-boxing: Music retrieval for the dj," in *Proceedings of the International Conference on Music Information Retrieval*, 2004, pp. 170–177.

[19] T. Nakano, J. Ogata, M. Goto, and Y. Hiraga, "A drum pattern retrieval method by voice percussion," *Database (Musical Instrument Sound)*, vol. 3, p. 1, 2004.

[20] A. Hazan, "Towards automatic transcription of expressive oral percussive performances," in *Proceedings of the 10th international conference on Intelligent user interfaces*, 2005, pp. 296–298.

[21] E. Sinyor, C. M. Rebecca, D. Mcennis, and I. Fujinaga, "Beatbox classification using ace," in *Proceedings of the International Conference on Music Information Retrieval*. Citeseer, 2005.

[22] D. Stowell and M. D. Plumbley, "Delayed decision-making in real-time beatbox percussion classification," *Journal of New Music Research*, vol. 39, no. 3, pp. 203–213, 2010.

[23] G. Peeters, B. L. Giordano, P. Susini, N. Misdariis, and S. McAdams, "The timbre toolbox: Extracting audio descriptors from musical signals," *The Journal of the Acoustical Society of America*, vol. 130, no. 5, pp. 2902–2916, 2011.

[24] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[25] B. Picart, S. Brognaux, and S. Dupont, "Analysis and automatic recognition of human beatbox sounds: A comparative study," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 4255–4259.

[26] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[27] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

[28] L. Nanni, G. Maguolo, and M. Paci, "Data augmentation approaches for improving animal audio classification," *Ecological Informatics*, vol. 57, p. 101084, 2020.

[29] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 558–567.

[30] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" *Advances in neural information processing systems*, vol. 31, 2018.

[31] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, H. Boyer, O. Mayor, G. Roma Trepat, J. Salamon, J. R. Zapata González, X. Serra *et al.*, "Essentia: An audio analysis library for music information retrieval," in *Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil.[place unknown]: ISMIR; 2013. p. 493-8.* International Society for Music Information Retrieval (ISMIR), 2013.

[32] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[33] E. Demirel, S. Ahlbäck, and S. Dixon, "Automatic lyrics transcription using dilated convolutional neural networks with self-attention," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020.

[34] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, "A compact model for speaker-adaptive training," in *Proceedings of the Fourth International Conference on Spoken Language Processing*. IEEE, 1996.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[36] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

A DATA-DRIVEN METHODOLOGY FOR CONSIDERING FEASIBILITY AND PAIRWISE LIKELIHOOD IN DEEP LEARNING BASED GUITAR TABLATURE TRANSCRIPTION SYSTEMS

Frank Cwitkowitz†
University of Rochester
fcwitkow@ur.rochester.edu

Jonathan Driedger
Chordify
jonathan@chordify.net

Zhiyao Duan
University of Rochester
zhiyao.duan@rochester.edu

ABSTRACT

Guitar tablature transcription is an important but understudied problem within the field of music information retrieval. Traditional signal processing approaches offer only limited performance on the task, and there is little acoustic data with transcription labels for training machine learning models. However, guitar transcription labels alone are more widely available in the form of tablature, which is commonly shared among guitarists online. In this work, a collection of symbolic tablature is leveraged to estimate the pairwise likelihood of notes on the guitar. The output layer of a baseline tablature transcription model is reformulated, such that an inhibition loss can be incorporated to discourage the co-activation of unlikely note pairs. This naturally enforces playability constraints for guitar, and yields tablature which is more consistent with the symbolic data used to estimate pairwise likelihoods. With this methodology, we show that symbolic tablature can be used to shape the distribution of a tablature transcription model's predictions, even when little acoustic data is available.

1. INTRODUCTION

Automatic Music Transcription (AMT) is a well-known task within the Music Information Retrieval (MIR) community dealing with the estimation of note content within a music signal [1]. Guitar tablature transcription refers to the specific problem of identifying all of the notes within a solo guitar recording and assigning the string that was used to play them. The task represents the determination of not only what was played, but also how it was played on the instrument. This information is necessary to realize tablature, a type of prescriptive notation for stringed instruments where fret numbers are superimposed atop lines representing each string. The fret numbers correspond to the notes that are to be played for a specific piece.

The guitar is a very popular musical instrument with users spanning all skill levels. The value of knowing how a piece was played on guitar is immeasurable for the vast community of guitarists learning to play the instrument. Guitar has a relatively low barrier to entry w.r.t. music theory knowledge, and many guitarists use guitar tablature instead of standard staff notation. Even more experienced players often prefer tablature for storing and communicating guitar-specific music ideas due to the intuitiveness and simplicity. Tablature is also widely shared across the internet through primarily user-curated websites such as *Ulimate-Guitar* [1] .

Despite the popularity of guitar, the instrument has received considerably less attention when it comes to music transcription. The main obstacles stem from a lack of audio recordings with transcription labels, or acoustic data, capturing the exceeding variability of the instrument. The guitar has many expressive dimensions such as the plucking style, plucking location, the use of embellishments [2] , etc. These and many other factors can affect the audio. Standard guitars do not have a digital interface, and expensive manual processing is required to obtain qualitative note labels. Without large datasets capturing the breadth of the intrument, it is very difficult to train reliable models and avoid over-fitting.

Since a standard guitar has six independent strings, it is a polyphonic instrument. This means that guitar transcription carries all of the intrinsic challenges of polyphonic note transcription. One further challenge is the estimation of the string on which each note was played. The pitch ranges of adjacent strings have a significant overlap, with their lowest pitches being only 4-5 semitones apart. The timbral cues of different strings are very subtle, and it is difficult to learn these without a lot of data.

There have been several attempts to realize systems which can transcribe solo guitar audio into tablature. Often, the approaches consist of two-stage systems, which first estimate pitch salience and then map pitch estimates to the guitar. Many works employ a basic signal processing pipeline, *e.g.* [2–4], whereby some form of spectral analysis or peak-picking is carried out. Some approaches attempt to estimate the string of detected pitches based off of inharmonicity measurements [5], which vary across strings. Others employ graph-search algorithms to find the optimal path through a list of fingerings for the observed pitches [6–8]. Often, to determine if a fingering or a transition is optimal or even feasible, these approaches have relied on rule-based procedures with hand-crafted weightings.

† Main work completed as a research intern at Chordify.

Copyright: © 2022 Frank Cwitkowitz et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[1] https://www.ultimate-guitar.com/

[2] Examples include bends, slides, hammer-ons/offs, vibrato, etc.

More recently, machine learning has become a popular strategy for guitar tablature transcription. Several works train Hidden Markov Models (HMMs) to model the transition between fingerings and chords [9, 10]. Other works perform classification to estimate, among other parameters, the string associated with each note using a collection of hand-crafted features [4], or the use of a Deep Belief Network (DBN) to produce pitch estimates [11]. Bayesian classification has also been proposed [12–14] to estimate the string and fret of notes.

Convolutional Neural Network (CNN) based approaches have been proposed [15, 16] to perform the task of pitch estimation and tablature arrangement jointly. These models benefit from being able to learn features directly from acoustic data, and tend to generalize much more effectively to real-world data. Our main contributions stem from a simple observation: the output layer formulation of these CNNs is prone to falsely producing tablature with duplicated pitches. This is because the output layer formulation represents six independent classification problems, *i.e.*, one per string. While the output neurons of each softmax group implicitly share information from previous layers of the network, they do not explicitly communicate when determining which class to choose. In contrast, the fretting of one string is highly correlated to other string frettings. This is related to what pitch intervals are likely to be played at different locations as well as bio-mechanical feasibility.

In order to incorporate the knowledge of likely fingerings and feasibility, we propose a new output layer formulation for guitar tablature transcription models [3]. In the new formulation, a novel inhibition objective is applied during training to discourage the concurrent activation of unlikely or infeasible fingerings. The inhibition weights are derived from the likelihood of co-occurrence for each pair of notes on the guitar, which is estimated using DadaGP [17], a large dataset of guitar tablature. The proposed output layer formulation essentially learns a language model without requiring acoustic data. We directly compare the new formulation to the previous formulation, and show that its predictions more closely match the distribution of DadaGP.

2. PROPOSED METHOD

Given a collection of symbolic tablature data, the pairwise likelihood of each string and fret (S/F) combination can be estimated. The new output layer formulation is amenable to training with the pairwise likelihoods through a pairwise inhibition loss. As a result, the predictions of the new output layer more closely match the distribution of the tablature within the collection. Assuming a preponderance of the tablature is playable, the generated tablature will naturally be more feasible to play. In the following sections, we introduce the baseline CNN architecture used in this work, and discuss the new output layer formulation in more detail, the process of estimating the pairwise likelihood of each S/F combination, and the proposed inhibition loss.

[3] All code is available at `https://github.com/cwitkowitz/guitar-transcription-with-inhibition`.

2.1 Baseline Model

We employ TabCNN [16] as our baseline model for guitar tablature transcription, leaving most of the original design choices largely unchanged. TabCNN is a simple CNN which processes Constant-Q Transform (CQT) frames and produces sets of fret class predictions. It was designed to be compatible with real-time processing, so it is relatively lightweight and operates on multiple CQT frames in order to make one set of predictions. The model comprised three 2D convolutional layers with ReLU activations, followed by a max pooling layer, a fully-connected layer with ReLU activation, and finally the output layer discussed in the following section. During training, dropout is applied after the max pooling layer and directly before the output layer. The network is trained using AdaDelta optimizer with an initial learning rate of 1.0. We implemented the model from scratch in PyTorch, using all of the same hyperparameters as in the original paper for feature extraction and the model architecture. We also insert a uni-directional long short-term memory (LSTM) [18] layer before the output layer. This is a simple modification which results in a relatively significant improvement (see Sec. 4), without disrupting the real-time processing capacity of TabCNN.

2.2 Output Layer Formulation

The output layer of TabCNN is a fully-connected layer with one softmax activation for each string. The output neurons represent all combinations of string $s \in \{1, ..., 6\}$ and fret class $f \in \{-1, 0, 1, ..., F\}$, where $f = -1$ represents a class for silence, $f = 0$ represents the open string, and F is the total number of frets supported. In the standard model, the softmax operations are applied independently to the neurons associated with each string. This results in probability activations $z_{s,f,n}$ for each frame n where $z_{s,f,n} > 0 \ \forall s, f, n$ and $\sum_{f=-1}^{F} z_{s,f,n} = 1, \ \forall s, n$. The loss for a track is then computed by summing the categorical cross entropy for each string group and averaging across the N total frames. This can be expressed as

$$L_{CCE} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{s=1}^{6} \log \left(z_{s,f',n} \right), \tag{1}$$

where $z_{s,f',n}$ is the activation corresponding to the ground truth fret f' for string s during frame n. Inference then consists of choosing the frets with the highest activation, resulting in six predictions for each frame. Using a binary representation the predictions can be written as

$$y_{s,f,n} = \mathcal{I}(\underset{f}{\operatorname{argmax}}\{z_{s,f,n}\} = f), \tag{2}$$

where $\mathcal{I}(\cdot)$ is the indicator function. We refer to this formulation as the 6D Softmax formulation. One advantage of this formulation is that the model is not capable of generating invalid predictions, *i.e.*, only one note can be chosen at maximum for each string. While this property is highly desirable, the six softmax activations act independent of one another, causing the model to treat transcription as six somewhat independent classification tasks. In practice, the ground truth S/F combinations making up a fingering arrangement are highly correlated at all times. This

is due to both the physical limitations of what a human hand can play as well as musical motivations that make it unlikely to play certain pitches at the same time.

In contrast, we propose to formulate the output layer as representing a binary classification problem for each fret of each string. In this way, the likelihood of each S/F combination being active is independently computed using a sigmoid activation. As such, we refer to the new formulation as the Logistic formulation. This formulation allows us to expand the loss function to include an inhibition objective to discourage the co-activation of certain pairs of S/F combinations. This additional objective would otherwise conflict with the 6D Softmax formulation, due to normalization of the activations in the softmax function.

The new loss is computed by summing the binary cross entropy across each fret class for each string and again averaging across all frames. For convenience, let us now fold string and fret into a single variable $c \in \{1, ..., C\}$ representing each distinct S/F combination, where $C = 6 \times (F + 2)$. The loss can then be expressed as

$$L_{BCE} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} t_{c,n}\log\left(z_{c,n}\right) + \left(1 - t_{c,n}\right)\log\left(1 - z_{c,n}\right), \quad (3)$$

where $t_{c,n}$ is a binary number indicating whether there is a positive class label in the ground-truth at the corresponding S/F combination c at frame n. This new output layer formulation is similar to that of standard piano transcription models [19]. However, we still use Equation (2) to obtain the final predictions, rather than considering each activation above a certain threshold a positive prediction. This is because, ultimately, we can still only choose one fret class per string. Note that the inhibition objective is applied during training and is unaffected by inference.

2.3 Estimating Pairwise Likelihood

Certain S/F combinations on the guitar have a very low chance of being played at the same time. These can include S/F combinations located on the same string, S/F combinations with the same pitch, S/F combinations which are far apart, or simply musically uncommon S/F combinations. In order to incorporate these considerations into the Logistic formulation, we estimate the likelihood of co-occurrence for all S/F combination pairs, to inform a novel training objective for pairwise S/F combination inhibition.

The pairwise likelihood of two S/F combinations c_i and c_j can be estimated using an arbitrary collection of symbolic tablature data. Here we define symbolic tablature data as one-hot encoded annotations for each string of the guitar at the frame-level. Given the symbolic tablature for a single track, we compute the intersection over union (IoU) of frame-level occurrences for all pairs of S/F combinations that co-occur in at least one frame. Mathematically, the intersection of a pair is defined as the number of frames where both combinations occur concurrently:

$$inter(i,j) = \sum_{n=1}^{N} t_{c_i,n} \wedge t_{c_j,n}. \quad (4)$$

The union is defined as a the number of frames where either of the combinations occur.

$$union(i,j) = \sum_{n=1}^{N} t_{c_i,n} \vee t_{c_j,n}. \quad (5)$$

Let $\mathcal{T}'(i,j)$ be the set of tracks where c_i and c_j, independently, each occur in at least one frame. Then, the IoU of the pair is averaged across these valid tracks:

$$IoU(i,j) = \frac{1}{|\mathcal{T}'(i,j)|}\sum_{t\in\mathcal{T}'(i,j)}\frac{inter(i,j)_t}{union(i,j)_t}, \quad (6)$$

where $|\mathcal{T}'(i,j)|$ is the cardinality of $\mathcal{T}'(i,j)$. Note that this is only valid for pairs where $|\mathcal{T}'(i,j)| > 0$. All pairs where $|\mathcal{T}'(i,j)| = 0$ receive $IoU(i,j) = 0$.

The final pairwise likelihoods are stored in a symmetric matrix, ordered by string and fret on both axes. As a result of Equation (6), the likelihood of a pair co-occurring with itself is always 1, and the likelihood of pairs within the same string co-occurring is always 0. This is convenient for the next step, as self-occurrence will never be inhibited, whereas same-string-occurrence will be maximally inhibited. The pairwise likelihood for the rest of the S/F combinations with differing strings will fall somewhere in between 0 and 1 (inclusive). See Fig. 1 for an example of a pairwise likelihood matrix.

2.4 Inhibition Loss

In order to apply the estimated pairwise likelihoods to the problem of guitar tablature transcription, we introduce a new loss term for inhibiting the co-activation of unlikely pairs. We refer to this as the inhibition loss L_{inh}. The inhibition loss requires that each pair of S/F combinations receive an inhibition weight $w(c_i, c_j)$ between 0 and 1, indicating how much to penalize the model for producing high activations for the combinations in the pair in a single frame. The inhibition weights here are chosen to be the complement of the pairwise likelihood (IoU) estimated in the previous step:

$$w(c_i, c_j) = (1 - IoU(i,j))^b, \quad (7)$$

where b is a parameter which boosts the effective pairwise likelihood by pushing the mass of the inhibition weights closer to 0. Many S/F combinations which tend to co-occur do not necessarily have a high likelihood of occurring together, relative to the amount of times they occur separately. This can make the estimated pairwise likelihood for many combinations small, leading to a high inhibition weight. Since we do not wish to discourage the activations of pairs which commonly co-occur, it can be helpful to boost the pairwise likelihood in this way. Empirically, we found that $b = 2^7$ produced weights with nice contrast between common pairs and uncommon pairs. Boosting can also be thought of as computing the joint probability of not observing c_i and c_j across b total frames, ≈ 3 seconds here. Given the inhibition weights, the inhibition loss for a sequence of N frame is computed as

$$L_{inh} = \frac{1}{2N}\sum_{n=1}^{N}\sum_{i=1}^{C}\sum_{j=1}^{C} z_{c_i,n} z_{c_j,n} w(c_i, c_j). \quad (8)$$

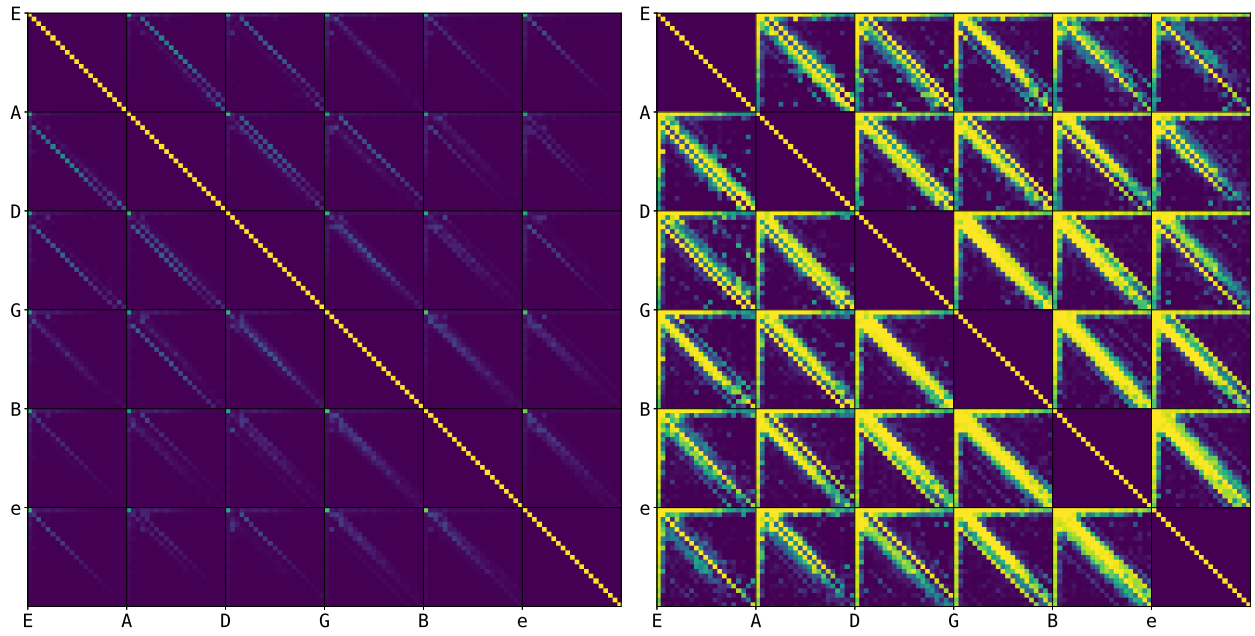


Figure 1. Visualization of estimated pairwise likelihood for all S/F combinations, ordered row- and column-wise by string s and fret f, computed using DadaGP [17] with $b = 1$ (left) and $b = 2^7$ (right). Grid lines are overlayed atop the string boundaries. The likelihoods range from 0 (dark purple) to 1 (bright yellow), where 1 represents pairs which always co-occur (*e.g.,* identical pairs on diagonal) and 0 represents pairs which never co-occur (*e.g.,* non-identical same-string pairs).

Here, the product for every combination of activations produced by the model is taken and scaled by the appropriate inhibition weight, and the result is subsequently summed over all combinations. Since both permutations of each S/F combination are included in the summation, we divide by two to remove redundancy. The total loss then becomes

$$L_{total} = L_{BCE} + \lambda L_{inh}, \tag{9}$$

where λ is a scaling term for balancing the two terms.

3. EXPERIMENTAL SETUP

In order to evaluate the efficacy of the proposed output layer formulation, we use it within TabCNN [16]. We compare it to the 6D Softmax layer, and experiment with several variations to study the effect of inhibition.

3.1 Datasets

We use two datasets for our experiments. GuitarSet [20] is used to train, validate, and test our models, and DadaGP [17] is used to compute weights for the inhibition loss applied during training and employed as an evaluation metric. We briefly introduce the datasets in the following sections.

3.1.1 GuitarSet

GuitarSet [20] is a guitar transcription dataset comprising roughly three hours of acoustic guitar audio. It contains various types of annotations, including pitch and note annotations with string labels. The provided labels were obtained by employing monophonic pitch tracking on the independent audio of each string, recorded with a hexaphonic pickup mounted to the guitar. The dataset features six guitarists playing two unique interpretations over 30 different chord progressions from various keys, resulting in 360 distinct tracks. Each track is approximately 1312.7 frames or 30.5 seconds on average. GuitarSet is used within a six-fold cross-validation schema, with the dataset splits representing the tracks produced by each respective guitarist. We limit training to only four splits during each fold, holding out one split for validation and one split for testing.

3.1.2 DadaGP

DadaGP [17] is a large collection of symbolic tablature encoded using the proprietary *GuitarPro* file format. The dataset features many popular songs from a variety of artists and spanning many musical styles, with a bias toward rock and metal music. The *GuitarPro* file format can store the transcription of multiple musical voices as tracks in a single file. Tracks corresponding to guitars have note labels which are always associated with an S/F combination.

We process all tracks corresponding to guitars in standard tuning within the *GuitarPro* files, ignoring duplicate [4] files. Note that many files include more than one guitar track. Using the Python package *PyGuitarPro* [21], we carry out a series of steps [5] where we assign an onset and offset to each note in the track to obtain tablatures in *JAMS* format [22]. Ultimately, we end up with 33967 pieces of symbolic tablature, most of which comes from full-length

[4] A duplicate is defined as having a preexisting file name or the "copy" tag in the file name. In cases where there are duplicates with alternate file extensions, we keep the one with the more recent *GuitarPro* version.

[5] Due to the many complexities of *GuitarPro*, we refer interested readers to the code for more information about these steps. It is also worth mentioning here that we treat slides and hammer-ons/offs as two separate notes, and that for bends we only retain the original note.

songs. We use these to estimate pairwise likelihoods.

3.2 Metrics

We compute all of the same metrics [6] as in [16]. These include precision, recall, and f-measure for all frames of tablature and the equivalent string-agnostic multipitch, as well as the tablature disambiguation rate (TDR), which indicates how well the model maps pitches to their respective strings. We also report the average inhibition losses L_{inh} and $L_{inh}{}^{+}$ using the standard and boosted inhibition weights, respectively. The inhibition losses serve as a proxy for how well the predictions of the model match the distribution of the symbolic tablature used to estimate the pairwise likelihoods. The lower the losses, the smaller the presence of inhibitory pairs in the predictions. Since the pairwise likelihoods were estimated using a large collection of real-world data, the inhibition losses also serve as a proxy for the feasibility of the predicted tablature. Note that the inhibition losses are calculated on the final predictions of a model. This is in contrast to the inhibition loss applied during training, which is computed on the activations after the sigmoid operation.

Additionally, we note that the inhibition weights tend to be very high for pairs on different strings corresponding to the same exact pitch $p \in \{1, ..., P\}$, where P is the total number of unique pitches. A duplicate pitch error is a common mistake made by the network, since the acoustic profile of the same pitch on different strings is very similar. We report the average number of duplicate pitch errors per track, where the duplicate pitch error count $E_{d.p.}$ for a track with N frames is expressed as

$$E_{d.p.} = \sum_{n=1}^{N} \sum_{p=1}^{P} \max(0, m_{p,n}^{(y)} - m_{p,n}^{(t)}), \qquad (10)$$

where $m_{p,n}^{(y)} \in \{0, \ldots, 5\}$ and $m_{p,n}^{(t)} \in \{0, \ldots, 5\}$ are the number of string-wise model predictions and ground-truth targets, respectively, at frame n corresponding to pitch p. Since it is extremely rare that a guitarist duplicates the same pitch in practice, this error count is a more explicit way of telling if the inhibition objective is effective. We also report the average number of false alarm errors,

$$E_{f.a.} = \sum_{n=1}^{N} \sum_{s=1}^{6} \sum_{f=0}^{F} y_{s,f,n} \wedge \neg\, t_{s,f,n}, \qquad (11)$$

indicating more generally all predictions made by the model which do not occur in the ground-truth. This allows us to compare the number of duplicate pitch errors to the total number of false alarm errors in each experiment. Note that here we do not consider incorrect silence predictions ($f = -1$) to be false alarm errors.

3.3 Experiments

We conduct a series of experiments to observe how the proposed output layer formulation compares to the baseline 6D Softmax formulation. All of our experiments are trained and evaluated on GuitarSet following the six-fold cross-validation schema laid out in [16]. There is one major difference in that we hold out one extra dataset split for validation. This means that per fold, only four splits are used for training, while the other two are used for validation and evaluation, respectively. Within each fold, the criterion for choosing the model checkpoint to evaluate on the evaluation split is the checkpoint with the maximum tablature f-measure on the validation split.

The purpose of first few experiments is to verify our reproduction of TabCNN and our experimental setup, and to investigate the difference in performance when using a validation set. The models in these experiments are trained with a batch size of 30 for 10000 iterations, where 200 consecutive frame groups within each track in the training set are sampled per iteration. Experiment (1) features our reproduction of TabCNN, with the standard 6D Softmax output layer formulation. In order to try to match the original TabCNN results as closely as possible, we do not perform validation in this experiment, and simply evaluate the final models within each fold at 10000 iterations. Next, in experiment (2), we run the same experiment but with the validation methology outlined above. All of the remaining experiments utilize the same validation methodology.

The purpose of the remaining experiments is to compare the 6D Softmax and Logistic output layer formulations directly. As discussed in Sec. 2.1, an LSTM is inserted before the output layer of each model in these experiments as a simple improvement, and to observe what happens to the proposed metrics when a simple language model is added. In order to create a balance between the sequence length and the batch size when training the models with the LSTM, we also modify the training hyperparameters such that we train with a batch size of 50 with a sequence length of 125 frame groups for 50000 iterations. We increase the amount of training iterations to account for the additional model complexity of the LSTM.

Experiment (3) is the same as Experiment (2), except for the insertion of the LSTM layer and the new training hyperparameters. Experiment (4) features the Logistic output layer formulation detailed in Sec. 2.2, but with an ineffective inhibition objective, *i.e.* $\lambda = 0$. In the remaining experiments, $\lambda \neq 0$, and various different matrices of inhibition weights are employed. In Experiment (5), a set of weights which only cover the hard string constraints is used for the inhibition loss. The weights can be expressed as the following:

$$w(c_i, c_j) = \begin{cases} 1 & \text{if } |i-j| < (F+2) \text{ and } i \neq j \\ 0 & \text{otherwise} \end{cases}. \qquad (12)$$

This matrix of weights inhibits all of the pairs which are intrinsically impossible with the 6D Softmax formulation. A visualization of the complement to these weights is provided in Fig. 2. All other experiments employ matrices estimated using DadaGP [17] with the procedure detailed in Sec. 2.3. The standard inhibition weights ($DadaGP$) are used in Experiment (6). Experiment (7) and (8) use inhibition weights boosted with $b = 2^7$ ($DadaGP^{+}$). The

[6] The original metrics were computed across all frames of all tracks in a track-agnostic manner, whereas we compute the metrics across all frames of each track and average results across all tracks. This is done to weight the influence of every track in the dataset equally.

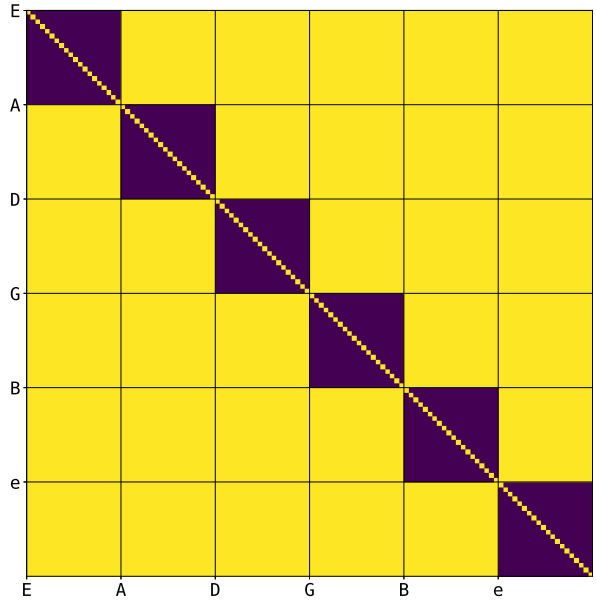


Figure 2. Complement of the inhibition weights which enforce hard string constraints, described by Equation (12).

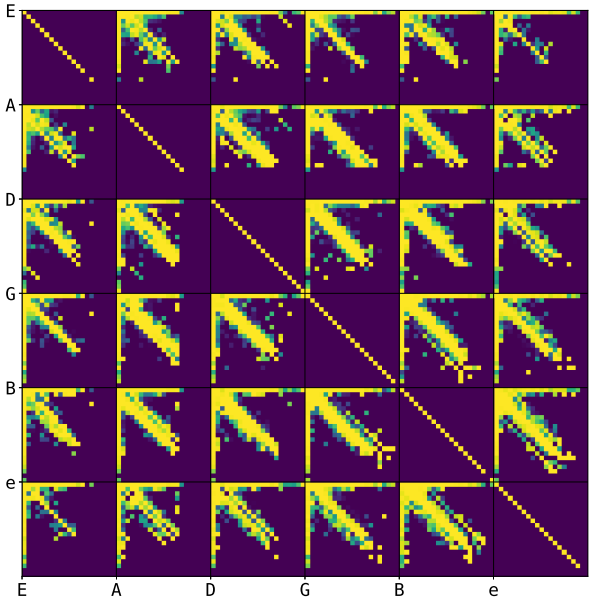


Figure 3. Estimated pairwise likelihoods for all S/F combinations computed using GuitarSet [20] with $b = 2^7$.

estimated pairwise likelihoods corresponding to each set of weights are presented in Fig. 1. Experiments (4-7) use an inhibition loss multiplier of $\lambda = 1$, whereas Experiment (8) uses $\lambda = 10$. These experiments make up a sort of ablation study w.r.t. the design choices of the inhibition objective within the Logistic formulation.

4. RESULTS & DISCUSSION

4.1 Transcription

The transcription results for the experiments outlined in Sec. 3.3 are presented in Table 1. The results of Experiment (1), our reproduction of TabCNN [16], are almost identical to what was originally reported. Regarding Experiment (2), validation lowers transcription performance slightly, as expected, but to a surprisingly small degree. This validates our decision to trade an extra dataset split for a more justifiable selection criterion. In Experiment (3), insertion of the LSTM significantly improves tablature transcription performance, mainly by increasing recall.

Without the inhibition objective, the logistic formulation in Experiment (4) yields essentially the same performance as the 6D Softmax formulation in Experiment (3). When using the inhibition objective with the string constrains in Experiment (5), the overall tablature transcription performance improves slightly. Since the weights only discourage activations on the same string from co-occurring, similar to the 6D Softmax formulation, this is an interesting result. With the $DadaGP$ weights in Experiment (6), the inhibition objective lowers performance slightly by decreasing tablature and multipitch recall. This is not surprising, since the weights are very strict and inhibit almost everything besides perfect 4^{th}/5^{th} intervals. With the boosted weights $DadaGP^{+}$ in Experiments (7-8), recall improves and precision drops slightly for tablature and multipitch. Finally, the stronger inhibition objective ($\lambda = 10$) in Experiment (8) lowers overall performance for multipitch estimation, while maintaining roughly the same tablature transcription performance w.r.t. Experiment (7).

The lack of an increase in tablature performance when using the inhibition objective can most likely be attributed to the small size of GuitarSet [20] and the presence of some noisy labels. Using the procedure detailed in Sec. 2.3, pairwise likelihoods were estimated from GuitarSet [20] for analysis. These are illustrated in Fig. 3. Clearly, there are co-occurrences for pairs which are highly inhibited by the $DadaGP$ and $DadaGP^{+}$ weights. Upon inspection of the tracks which produced these artifacts, we found some instances of duplicate pitch errors and octave errors in the annotations. It could be that the inhibition variants actually avoided making these types of predictions, but received a lower score due to the likely annotation errors. Furthermore, the rock and metal bias in DadaGP [17] may have skewed the distribution of model predictions in the relevant models away from the distribution of GuitarSet [20], which contains genres such as Jazz and Bossa Nova.

4.2 Loss & Errors

The distribution and error measurements for the experiments outlined in Sec. 3.3 are presented in Table 2. Some key observations can be made about these results. First, the standard inhibition loss L_{inh} remains relatively consistent across experiments. As noted earlier, the $DadaGP$ weights are very strict and inhibit most pairs, so the inhibition objective can conflict with the transcription objective. For this reason, it makes sense that among Experiments (3-8), Experiment (6) had the most influence on this metric, given that it used L_{inh} for training. We do observe a significant decrease in ${L_{inh}}^{+}$ for Experiments (7-8), which

Tablature Layer	p_{tab}	r_{tab}	f_{tab}	p_{pitch}	r_{pitch}	f_{pitch}	TDR
(1) Reproduction	0.809	0.692	0.742	0.910	0.762	0.825	0.903
(2) Reproduction w/ Val.	0.775	0.696	0.730	**0.895**	0.781	0.830	0.886
(3) Reproduction w/ Val./Rec.	0.783	0.757	0.768	0.879	0.835	0.854	0.905
(4) Logistic (No Inhibition)	0.782	0.757	0.767	0.878	0.836	0.854	0.902
(5) Logistic w/ String Constraints	**0.789**	**0.761**	**0.773**	0.881	**0.836**	**0.856**	**0.907**
(6) Logistic w/ DadaGP	0.787	0.743	0.763	0.880	0.821	0.847	0.902
(7) Logistic w/ DadaGP$^+$	0.782	0.754	0.766	0.876	0.833	0.852	0.902
(8) Logistic w/ DadaGP$^+$$(\lambda = 10)$	0.781	0.755	0.766	0.867	0.829	0.845	0.907

Table 1. Average six-fold cross validation results on GuitarSet [20] for transcription metrics. Bold values indicate the highest observed result for each metric across experiments which follow the validation methodology outline in Sec. 3.3 (*i.e.,* Experiments (2-8)). The break separates experiments without recurrence and experiments which used an LSTM.

Tablature Layer	L_{inh}	$L_{inh}{}^+$	$E_{d.p.}$	$E_{f.a.}$
(1) Reproduction	8.87	0.132	21.4	359.8
(2) Reproduction w/ Val.	**9.01**	0.152	34.2	**442.5**
(3) Reproduction w/ Val./Rec.	9.27	0.158	24.3	489.6
(4) Logistic (No Inhibition)	9.27	0.154	20.0	503.3
(5) Logistic w/ String Constraints	9.25	0.155	19.5	485.8
(6) Logistic w/ DadaGP	9.19	0.147	12.0	481.8
(7) Logistic w/ DadaGP$^+$	9.25	0.143	13.8	496.6
(8) Logistic w/ DadaGP$^+$$(\lambda = 10)$	9.26	**0.132**	**10.6**	504.6

Table 2. Average six-fold cross validation results on GuitarSet [20] for distribution and error metrics. Bold values indicate the lowest observed result for each metric across experiments which follow the validation methodology outline in Sec. 3.3 (*i.e.,* Experiments (2-8)). The break separates experiments without recurrence and experiments which used an LSTM.

train with the boosted inhibition loss. In contrast to L_{inh}, even when $L_{inh}{}^+$ continues to go down, the transcription performance improves for Experiments (6-8). This agrees our hypothesis that the boosted inhibition weights are more suitable for use with the transcription objective.

We also notice that, despite a significant increase in Experiment (2) and subsequently Experiment (3), the average number of false alarm errors $E_{f.a.}$ remains relatively consistent across Experiments (3-8). It does seem to improve slightly with inhibition using string constraints and the $DadaGP$ weights. The average number of duplicate pitch errors $E_{d.p.}$ improves significantly in Experiments (6-8). As discussed before, there are cases of duplicate pitches in the ground-truth of GuitarSet. The reason why the duplicate pitch error count is not lower may be because the models are trained to produce duplicate pitch predictions in some scenarios. Overall, we argue that the lower $E_{d.p.}$ and $L_{inh}{}^+$ suggests that models trained with $DadaGP$ and $DadaGP^+$ produce tablature which is more feasible to play and more consistent with DadaGP [17].

4.3 Future Work

Although the inhibition objective was presented here in the context of improving guitar tablature transcription, it has several other potential uses. Inhibition shapes the distribution of model predictions by inhibiting unlikely S/F pairs. This means that one can estimate the pairwise likelihoods using a curated distribution, *e.g.* a collection of tablature corresponding to a specific musician or genre. Similarly, collections based on playing difficulty could be curated to influence the model to produce tablature more suitable for specific users with varying proficiency.

Another usage could be within the context of tablature arrangement, where the inhibition objective could be applied to train a model to allocate a set of preexisting pitches to strings, such that the resulting fingerings are playable. This may even be useful in a two-stage approach to tablature transcription, where a generic multipitch estimation model feeds predictions into the arrangement system.

We also suggest several directions for improving the inhibition objective. First, one could explore various types of data augmentation for symbolic tablature. One example we refer to as capo augmentation, where a constant fret offset is added to all notes within a track. This could prevent issues related to a dataset's lack of fretboard coverage. Another interesting direction would be the inclusion of a temporal inhibition objective at the note-level. This could prevent a model from generating predictions which shift around the fretboard too much. Lastly, it would be interesting to investigate higher-order S/F relationships (*e.g.*, 3 notes or more), since the current method only takes pairwise relationships into account.

5. CONCLUSION

We propose a new output layer formulation for guitar tablature transcription which takes advantage of large collec-

tions of symbolic tablature data. The pairwise likelihood of concurrent activation for all possible notes on the guitar is estimated using a recently published dataset. The complement of the pairwise likelihood is used as a weight for an accompanying inhibitory objective during training. We compare the new formulation against the output layer formulation of a baseline tablature transcription model. The inhibition objective is shown to be effective in shaping the distribution of the output predictions and lowering the number of duplicate pitch errors. We also discuss alternative uses and future directions for the inhibition objective.

Acknowledgments

This work has been partially funded by the National Science Foundation grants IIS-1846184 and DGE-1922591.

6. REFERENCES

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[2] X. Fiss and A. Kwasinski, "Automatic real-time electric guitar audio transcription," in *Proceedings of ICASSP*, 2011.

[3] L. Alcabasa and N. Marcos, "Automatic guitar music transcription," in *Proceedings of the International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, 2012.

[4] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score and instrument-related parameters," in *Proceedings of DAFx*, 2014.

[5] I. Barbancho, L. J. Tardon, S. Sammartino, and A. M. Barbancho, "Inharmonicity-based method for the automatic generation of guitar tablature," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 20, no. 6, pp. 1857–1868, 2012.

[6] G. Burlet and I. Fujinaga, "Robotaba guitar tablature transcription framework," in *Proceedings of ISMIR*, 2013.

[7] K. Yazawa, D. Sakaue, K. Nagira, K. Itoyama, and H. G. Okuno, "Audio-based guitar tablature transcription using multipitch analysis and playability constraints," in *Proceedings of ICASSP*, 2013.

[8] K. Yazawa, K. Itoyama, and H. G. Okuno, "Automatic transcription of guitar tablature from audio signals in accordance with player's proficiency," in *Proceedings of ICASSP*, 2014.

[9] A. M. Barbancho, A. Klapuri, L. J. Tardon, and I. Barbancho, "Automatic transcription of guitar chords and fingering from audio," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 20, no. 3, pp. 915–921, 2011.

[10] G. Hori, H. Kameoka, and S. Sagayama, "Input-output HMM applied to automatic arrangement for guitars," *Information and Media Technologies*, vol. 8, no. 2, pp. 477–484, 2013.

[11] G. Burlet and A. Hindle, "Isolated guitar transcription using a deep belief network," *PeerJ Computer Science*, vol. 3, p. e109, 2017.

[12] J. Michelson, R. Stern, and T. Sullivan, "Automatic guitar tablature transcription from audio using inharmonicity regression and bayesian classification," *Journal of the Audio Engineering Society (AES)*, 2018.

[13] J. M. Hjerrild and M. G. Christensen, "Estimation of guitar string, fret and plucking position using parametric pitch estimation," in *Proceedings of ICASSP*, 2019.

[14] J. M. Hjerrild, S. Willemsen, and M. G. Christensen, "Physical models for fast estimation of guitar string, fret and plucking position," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019.

[15] E. J. Humphrey and J. P. Bello, "From music audio to chord tablature: Teaching deep convolutional networks to play guitar," in *Proceedings of ICASSP*, 2014.

[16] A. Wiggins and Y. Kim, "Guitar tablature estimation with a convolutional neural network," in *Proceedings of ISMIR*, 2019.

[17] P. Sarmento, A. Kumar, C. Carr, Z. Zukowski, M. Barthet, and Y.-H. Yang, "DadaGP: A dataset of tokenized GuitarPro songs for sequence models," in *Proceedings of ISMIR*, 2021.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proceedings of ISMIR*, 2018.

[20] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "GuitarSet: A dataset for guitar transcription," in *Proceedings of ISMIR*, 2018.

[21] S. Abakumov, "PyGuitarPro," [Online], available at: https://github.com/Perlence/PyGuitarPro.

[22] E. J. Humphrey, J. Salamon, O. Nieto, J. Forsyth, R. M. Bittner, and J. P. Bello, "JAMS: A JSON annotated music specification for reproducible MIR research," in *Proceedings of ISMIR*, 2014.

muco: A MUSIC COMPUTING LEARNING APPLICATION

Patricia Hu[1, 4], Oliver Hödl[1, 2], Peter Reichl[1], Fares Kayali[2], Iris Eibensteiner[3], Bernhard Taufner[3], Sigrid Schefer-Wenzl[3], and Igor Miladinovic[3]

[1]Cooperative Systems Research Group, University of Vienna, Austria firstname.lastname@univie.ac.at
[2]Centre for Teacher Education, University of Vienna, Austria firstname.lastname@univie.ac.at
[3]Software Design and Engineering Department, University of Applied Sciences Campus Vienna, Austria firstname.lastname@fh-campuswien.ac.at
[4]Institute of Computational Perception, Johannes Kepler University Linz, Austria patricia.hu@jku.at

ABSTRACT

Following a computational approach to music creation can serve as a transdisciplinary bridge between computer science, music and education by providing rich opportunities for teaching computing skills in interdisciplinary contexts. Making use of such learning contexts is one of the cornerstones of STEAM (STEM + Arts) education, a pedagogic approach focusing on integrating the arts and humanities into STEM fields. Despite numerous benefits reported in the literature, the number of STEAM programs integrated into formal learning contexts remains limited due to various integration challenges. In addressing these challenges, we developed and evaluated a gamified music computing learning application called muco, using the MVC software architecture along with React as the front-end, and Node and MariaDB as back-end technologies. The goal of the application is to make programming concepts more accessible and comprehensible to a broader public, and to simultaneously spark interest in music and music making. Making use of game mechanics, muco is conceptualised to teach computational thinking, and more specifically, introductory programming, in non-formal learning contexts. The usability of the application and impact of interdisciplinary learning is evaluated by means of both formative and summative testing. Results show that the application provides an engaging learning environment which encourages the exploration of both the computing and music domain.

1. INTRODUCTION

There are many parallels between musical and software artefacts. To a certain extent, music can be considered 'organised sound', containing hierarchical structures formed by musical events of different levels, each varying in their degree of abstraction [1, 2]. These inherent structural relationships and the hierarchical nature in and between musical patterns offer ample grounds to approach music in an algorithmic or computational manner [3, 4].

Copyright: © 2022 Hu P. et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

The idea of adopting an algorithmic approach to music and music making has in fact been explored across millennia and across cultures [5]. More recently, a number of researchers have also pointed to the potential of adopting an analytical approach to music to enrich existing didactic methods in either music or computer science disciplines [4, 6]. In the latter case in particular, a systematic and structured engagement with music can provide a rich pedagogic context for exploring the creative sides of computing [7, 8].

The field of computational thinking within educational research and practice is steadily growing, and an increasing number of countries worldwide are integrating programming education into their national school curricula [9, 10]. As suggested above, music and music education (involving various aspects of music such as understanding, analysis, composition etc.) provide many opportunities for teaching computational thinking and programming concepts in a creative context. This is especially true in the area of STEAM (STEM + Arts) education.

The muco web application builds on the premise of bridging two domains - computing and music - to explore and exploit interdisciplinary potentials for knowledge transfer and learning. The application is intended to be suitable for beginners of both fields, thus requiring no prior knowledge in either music or programming.

Likewise, striving to advance the field of STEAM education, the muco application is designed to be used in non-formal, non-structured pedagogic settings. To encourage learning persistence in such contexts, the application leverages the potential of gamification techniques in order to promote student engagement and motivation, both of which have been identified as key factors in any learning process [11]. To this end, the muco application features selected game elements.

The remainder of this work is structured as follows: after providing a succinct introduction to the relevant subject areas in section 2, section 3 illustrates the applied research approach. Next, section 4 presents the muco application from both a technical and content-related point of view. Section 5 then describes the evaluation of the application, with results being presented and discussed in section 6. Lastly, section 7 concludes with a summary of lessons learned and areas for future research.

On a final note, it should be noted that the results pre-

sented herein do not claim to be exhaustive. Rather, the present work strives to be explorative in nature, pursuing the overall goal of making computing disciplines more accessible and comprehensible to a broader audience, while simultaneously sparking interest in music and making music.

2. RELATED WORK

2.1 STEAM Education

STEAM (as an acronym of the included disciplines Science, Technology, Engineering, Arts and Mathematics) is a transdisciplinary pedagogic approach focusing on the incorporation of liberal arts and the humanities into classical STEM education. As an emerging educational practice, it is increasingly gaining importance and recognition within the pedagogic community [12–14].

The literature on STEAM education reports a wide range of positive outcomes, with multiple empirical studies demonstrating improved content acquisition and conceptual understanding of scientific topics [15–17], increased task- or skill-related proficiency [16,18,19] and overall improved motivation and attitudes towards a specific STEM discipline [8, 14, 16, 19]. Despite these positive effects, however, researchers have identified recurring issues such as a lack of shared understanding of the STEAM concept and differing opinions on the nature and role of arts integration [13].

Another challenge reported in the literature relates to practical obstacles hindering effective STEAM program implementation. In this context, four major problem sources have been identified by Herro et al. [20], which relate to adequate pacing and time allocation of STEAM teaching units, observed student struggles with self-directed learning, challenges related to content and discipline alignment as well as teacher collaboration, and issues resulting from inflexible school policies.

2.2 Computational Thinking and Music

The concept of computational thinking emerged in the 1950s and 1960s in the field of computer science as 'algorithmic thinking'. In the traditional sense, the term can be described as a habit of mind of designing and crafting useful and effective software programs [21].

Since then, a variety of definitions of the term have been proposed [9, 22], most of which involve the mental concepts of problem reformulation, abstraction and decomposition. Referring to the relation of computational thinking to programming, it is clear that both are not the same, but being able to think computationally does facilitate the transition of problem-solving methods to formalized programming languages [23]. Though the definition and taxonomy of computational thinking is still evolving [24], the concept in itself has been met with great enthusiasm by educators as well as academics, not least due to its contribution in attracting attention to the field of computer science [25,26].

Likewise, pedagogic scholars and practitioners from both computing and music fields have pointed to the educational potential of interdisciplinary overlaps. As indicated above, musical pieces and software programs mirror each other in a number of aspects. Building on the idea of approaching music in an analytic manner, a number of researchers have emphasized parallels not only between musical and software artefacts, but also between the thought processes involved in learning and making music and software [3, 4, 6–8].

2.3 Digital Education and Gamification

Digital learning refers to any learning framework or system involving the use of information and communication technologies (ICT), including electronic learning (e-Learning) and mobile learning (m-Learning) [27]. Gamification implies the incorporation of game thinking and game design techniques to conventionally non-game activities. In the educational context, this is often applied to increase student engagement and improve learning outcomes overall [28]. Making use of its resources and innovative technologies, digital learning offers an ideal environment for the integration of game elements and mechanics.

The idea of using gamification to enhance student engagement and motivation has been applied and studied in formal and informal education for some time [29]. Both digital learning and the application of gamification therein have also led to increased focus and emphasis on the concept of self-directedness or self-directed learning, which stresses the ability of the student to plan and manage resources independently and apply critical problem-solving [30].

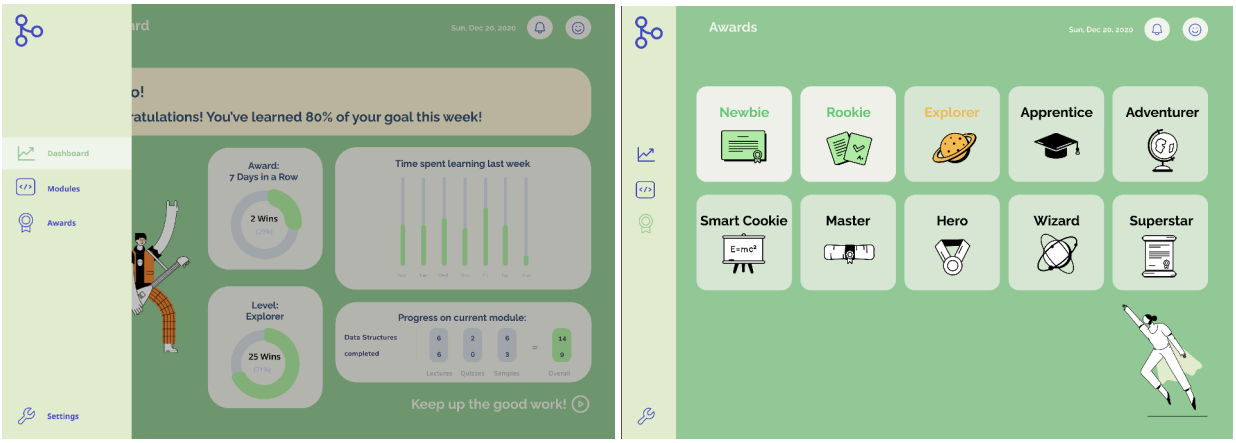
3. METHODOLOGY

Given the interdisciplinary nature of the present work, a mixed-methods approach was applied for the different steps involved in the design, implementation and evaluation of the `muco` learning application.

First, an iterative prototyping approach was followed for the design and implementation of the `muco` application, with each iteration realising a higher functional scope than the previous one. Next, the application was evaluated both qualitatively and quantitatively by means of formative and summative usability testing.

Formative testing was performed with an earlier prototype version of the `muco` application, specifically one which did not involve any gamification mechanics. These early-stage studies aimed at evaluating the general concept and learning content of the `muco` application and were conducted by means of semi-structured expert interviews. Interviews provide an appropriate context to ask open-ended questions on a broad range of topics which makes them suitable for gaining a deep and nuanced understanding of a problem [31].

Summative testing tends to focus on task measurements and related quantitative metrics, and is therefore more appropriate for more advanced development stages [32]. In the context of the present work, two summative usability studies were conducted to evaluate the final prototype version, which included game mechanics.



(a) (b)

Figure 1: (a) Dashboard (with open sidebar) and (b) Collectible Rewards in the muco application

4. MUCO APPLICATION

4.1 Design Rationale

A a starting point, a low-fidelity wireframe was created, which defined the fundamental structure and navigation of the application. Figure 1 presents excerpts from this wireframe. The design process was guided by Nielsen's *Ten Usability Heuristics* [33] and Shneiderman's *Eight Golden Rule for Interface Design* [34], both of which can be considered best practice principles in the realm of interface and usability design.

The most relevant functional requirements included the choice of a synthesis engine which supports the use of a general purpose language for teaching programming concepts and a browser-supported text-based editor serving as the development environment. Taking into account the design rational of conceptualising muco for use in non-formal, self-directed learning contexts, a rewards collection and dynamic module unlocking feature along with selected game elements were also included in the functional scope.

4.2 Technology Stack

The muco application is realised as a web application using React [1] in the Front-End, Node [2] and MariaDB [3] in the Back-End, and the Model-View-Controller (MVC) design pattern for the software architecture. The implementation of the muco application was preceded by the definition of key design and functional requirements, both of which formed the basis for the choice of application type, technology stack and software architecture.

With regards to the underlying synthesis engine, the Tone.js [4] library was chosen as it supports coding music in JavaScript, a high-level general purpose language that is both industry-relevant and beginner-friendly [35]. Tone.js is a web-based audio framework enabling interactive music making in the browser. The library provides basic objects and methods for creating low-level oscillators and synthesizers. In addition, the library also features higher-level monophonic and polyphonic instruments and samples, as well as methods for adding effects and controlling the scheduling of multiple audio events. For the realisation of a text-based development environment, both the Ace Editor [5] and CodePen pens [6] were used, with the former serving as read-only editors for presentation and illustration purposes and the latter providing an interactive development environment in which the user could try out (and run) new programming concepts.

4.3 Learning Modules

From a content and subject matter point of view, the main focus was centered on the learning modules, which were developed in two stages. In a first step, the structure for all modules was drafted as follows:

- *Theory* component: The first part of each learning module presents sound and music related concepts. They are intended to serve as theoretical constructs that can be used to illustrate programming concepts that are presented in the subsequent component.
- *Code* component: In the second part of each learning module, specific programming concepts are introduced. Each concept is explained and illustrated by means of a concrete code example that is displayed in an editor component. Each of the code examples also produces an audio output, which the user can trigger by running the code.
- *Try It Yourself* component: In the third part of each learning module, the user is encouraged to try out the previously introduced concept by him/herself and/or test him/herself on the content of the current module. To this end, this component provides either an interactive development environment, or a gamified test element.

[1] https://reactjs.org/ (accessed 31-10-21)
[2] https://nodejs.org (accessed 31-10-21)
[3] https://mariadb.org/ (accessed 31-10-21)
[4] https://tonejs.github.io/ (accessed 31-10-21)
[5] https://ace.c9.io/ (accessed 31-10-21)
[6] https://codepen.io/ (accessed 31-10-21)

Figure 2: The general structure of a learning module as illustrated by the first module, titled 'Learning By Playing'. The yellow area contains the *Theory*, the orange area the *Code*, and the blue area the *Try It Yourself* component.

Figure 2 presents the general structure described above, as illustrated by the introductory module titled 'Learning by Playing'.

In a second step, the actual structure of the learning contents and teaching objectives with regards to the music and programming concepts were defined and formulated. The learning content of the application was formalised in such a way to be suitable for beginners in either field, thus requiring neither musical nor programming prior knowledge.

Table 1 provides an overview of the learning modules provided in the application, including a list of the music and programming concepts covered in each module.

4.4 Game Mechanics

With the aim of increasing student engagement and motivation in self-directed learning contexts, a rewards collection and dynamic module unlocking feature were integrated into the `muco` application along with selected game elements. The game elements were implemented in the form of micro-assignments in the *Try It Yourself* components of each learning module, either as quizzes consisting of multiple-choice and/or single choice questions or sorting tasks in which the user is asked to bring an unsorted code script into the right order. To encourage the user to continue learning, these micro-assignments were linked to the module unlocking and rewards collection mechanism in such a way that requires the user to successfully pass an assignment in order to unlock a new module. In addition, the rewards were conceptualised to reflect different skill levels through corresponding badge titles.

5. EVALUATION

5.1 Formative Testing

Taking into account the interdisciplinary aspect of the `muco` application, domain experts from the field of music were selected as interview partners. To this end, five formally trained musicians were invited as study participants, among them two performing musicians, one music education student, one music teacher and one experimental media and sound artist. Except for one all participants were in their twenties (M27, M27, F22, M24, M35). Furthermore, except for the experimental musician, none had any prior programming knowledge or experience.

The interviews were carried out in a semi-structured manner and followed a conversational rather than inquisitorial communication style. A time frame of two to three hours was allocated for each interview to ensure sufficient time for both hands-on exploration and follow-up questions. At the beginning of each interview, a short introduction on the research context was given and informed consent was obtained regarding the collection and further analysis of the data acquired in the course of the study. Next, each participant was asked to introduce her/himself. This was followed by questions on expectations towards the `muco` application, which aimed at capturing the interviewees' first impression of the general concept. In particular, the purpose of these initial questions was to assess whether the interdisciplinary context explored in the application was comprehensible in itself, and whether the pedagogical approach stimulated curiosity and offered incentives for further exploration of the application. The next stage in the interview featured a hands-on exploration of the application in its then-current state, followed by questions on both the learning content and the perceived usability. The interview then concluded with a discussion on final reflections.

5.2 Summative Testing

Two summative studies were conducted to evaluate the final prototype version. Both studies incorporated two evaluation tools, that is, the System Usability Scale (SUS) questionnaire and a custom survey. The former is a standardised usability questionnaire designed for post-study scenarios, that is, after completion of a list of tasks in a test scenario. The questionnaire consists of ten items, each of which provides a five point Likert scale response option. It was selected as a research instrument because of its concise format and sufficient reliability for benchmark comparison purposes [32]. The latter was designed to gain a better understanding of certain aspects related to the content and use of the application and thus to substantiate (or qualify) the results of the previous formative usability study. Furthermore, it served as a tool for collecting demographic data.

Both studies took place in formal testing environments, that is, a lecture for non-beginner computer science students (n=28) and a research seminar for advanced students of educational sciences (n=4). A time frame of approx.

Module Title	**Sound and Music Concepts**	**Code Concepts**
Learning By Playing	Synthesizer, Oscillation, Sample, Modulation, Waveshaping, Timbre, Delays	Variable, Loop, Function, Function Body, Function Argument, Parameter, Callback
Sound and Waves	Sound wave, Oscillation, Sinusoid / Sine Wave, Oscillator	Variable, Variable Declaration and Variable Initialization, Object, Argument
Frequency and Pitch	Frequency, Hertz, Pitch, Concert Pitch	Variable Value, Data Types, Number, String, Method Chaining
Different Waveforms	Fundamental Frequency, Harmonicity, Overtones/ Partials, Primary Waveforms (Sine, Square, Triangle, Sawtooth)	Array, For Loop, Code Comments
Dynamics and Loudness	Dynamics, Loudness, Sound Power, Sound Intensity, Decibel	Introduction to Programming Best Practices

Table 1: Overview of learning modules provided in the `muco` application

one hour was allocated in both cases. Most participants in both test settings were in their twenties. Given their study background, all participants in the first test group had prior knowledge and experience in programming. In the second group, only one participant reported to have some basic understanding of programming concepts. Surprisingly, the number of participants who have had some form of formal music education was relatively high in both groups (39,3% and 50%, respectively). While gender balance was maintained in the second group, the ratio of female to male participants in the first group was 1:3.

At the beginning of each study, a short overview of the research context and study was given. This included a short illustration of the main structure of the `muco` application. Subsequently, a task list was presented, listing the tasks which participants were asked to perform prior to responding to the questionnaire and survey via Google Forms [7]. Before sending out the links to the application and Google Forms, informed consent was obtained with respect to the processing and analysis of the collected data. Participants were then given 30-45' for exploring the content and functionality of the application and for responding to the Google Forms. Depending on the respective field of study of each test group, the survey was subjected to slight modifications.

6. RESULTS

This section presents the results obtained by means of both formative and summative test procedures. Overall, the results of the formative studies, that is, the interviews conducted with domain experts from the musical domain, highlighted general curiosity and acclaim with respect to the interdisciplinary learning context, but revealed mixed responses with regards to the learning content. The results obtained by means of the two summative studies involving non-beginner computer science and educational sciences students revealed mixed results with regards to the overall learning and teaching concept and the learning content provided in the application. The user interface and experience was praised by study participants in both study formats. The following thematically groups and describes the results in further detail.

6.1 Response on Interdisciplinary Concept

In the formative setup, all interviewees expressed great curiosity about the interdisciplinary learning concept provided in the application. Despite the general enthusiasm, however, some misunderstanding on the intended use case and target user group of the application was perceived. However, this only concerned one participant.

In the summative setup, in the first group (formed of computer science students), the general concept of combining music and music related concepts with computer science and computational thinking in order to teach programming raised some doubts. Though most respondents praised the playfulness and the individual features of the application (the code sonification and the game mechanics, particularly), some respondents did question the exact use case of the learning application. In the second group, the students with a background in educational sciences were more welcoming of the general concept and particularly highlighted the *Try It Yourself* sections in each learning module and emphasised their importance in the context of learning new concepts. Participants in both groups praised the playful and creative approach and game mechanics featured in the application, with the code sorting tasks attracting the most acclaim.

6.2 Response on Learning Content

In the formative studies, the response on the structure and content of the learning modules was mixed, ranging from very positive to neutral to fairly negative attitudes. Two participants expressed strong interest and explicitly articulated they would want to continue using the application once it featured more learning modules. One participant reported feeling neutral towards the application, and the overall teaching concept. Of the remaining two partic-

[7] https://about.google/intl/forms/ (accessed 31-10-21)

ipants, one expressed having felt overwhelmed with the coding-related concepts, and therefore discouraged to use the application in an informal (i.e. self-directed) learning context. The other participant was equally unenthusiastic, but for the opposite reason, that is, the learning modules were perceived as too fragmented and incremental. It should be noted that this participant was the one having had prior programming knowledge.

The responses in the summative studies were likewise mixed, ranging from participants describing the content as advancing too slowly to test users complaining about the length and complexity of the modules. Both groups also included participants who explicitly praised the learning content, emphasising its step-by-step approach and the text-highlighting feature of important concepts.

6.3 Response on User Experience

In both the formative and summative tests, the responses on the user experience of the `muco` application were very positive, with many participants pointing out the appealing interface design. Despite these general positive remarks, however, a number of participants (over all study formats) expressed they would appreciate some form of visualisation as a learning aid for certain topics. In particular, sound waves were mentioned as useful and suitable means for visualisation purposes.

6.4 SUS Score

As mentioned earlier, the SUS was applied as a research instrument in the summative tests. For the first participant group (formed of computer science students), the overall SUS score obtained amounted to 70.8 (of 100). This is slightly above the general average score of 68 (at the 50th percentile), and can thus be considered 'acceptable' [36]. Note that for the second group (formed of students of educational sciences), the SUS score was not computed due to the small sample size (n=4). Fig. 3 presents the overall SUS score as broken down into its usability and learnability subscores for the first group.

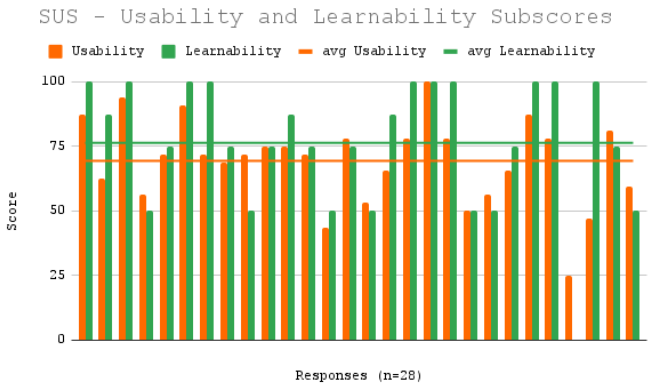


Figure 3: SUS subscores obtained in the formative study involving computer science students

6.5 Limitations

As with any other research method, expert interviews depict inherent weaknesses, not least due to their qualitative nature and the resulting risk of obtaining biased, subjective results. To address this shortcoming, the interviews conducted in the scope of the present work were complemented with two quantitative techniques, involving a standardised questionnaire and a broad-based survey. Despite these efforts, the total number of participants (that is, participants from all evaluation studies) remains rather small (N=37), thus limiting the significance of the results obtained.

Another limitation of this work concerns the characteristics of the test participants with respect to the following two aspects: First, regarding the knowledge and skill background of test participants, it must be emphasised that a large number of participants (i.e., those coming from a computer science study background) already had solid programming skills. Second, regarding the gender ratio, the gender imbalance encountered over all participants (across all evaluation studies) must be noted, with female participants accounting for only 27% of total participants.

On a final note, it should be emphasised that the results presented herein do not claim to be complete or exhaustive. Particularly, the concerns raised above necessitate the conduction of further usability studies with test groups which are both larger in size and more diverse with respect to both gender and each individual participants' background.

7. CONCLUSION

The present work focuses on the design and implementation of `muco`, a web-based music computing learning application which builds on the premise of combining music and music related concepts with computer science and computational thinking constructs to teach programming to beginners through coding music. Exploring and exploiting the potential of such interdisciplinary learning contexts is one of the principal considerations of STEAM education. Despite promising results of STEAM programs reported in the literature, researchers have likewise pointed to a number of integration challenges hindering large-scale implementation of STEAM-based curricula in school and classroom contexts [20]. In addressing these integration issues, the `muco` application was further conceptualised for use in non-formal (self-directed) learning contexts. To this end, the application leverages selected game mechanics to encourage learning persistence in such contexts [30].

Another issue commonly reported in the STEAM literature is that concerning the role and extent of arts integration [13]. Though not specifically addressed, in the course of conceptualising and defining the learning modules, a number of fundamental sound and music concepts were integrated were incorporated in an attempt to balance the roles of the disciplines considered.

The application was subjected to appropriate evaluation methods, involving both formative and summative usability tests depending on the level of maturity and functional scope of the respective development stage. Given the inter-

disciplinary nature of the learning concept featured in the application, test participants included domain experts from the fields of music, computer science and educational sciences. Formative testing was performed by means of expert interviews, whereas summative testing relied on the SUS standardised questionnaire and a custom survey as research tools. Summarising the results from both study formats, it can be said that the interdisciplinary learning context resulting from the combination of music and coding proves to be suitable for sparking student interest and motivation, and is therefore deemed worthwhile for further exploration. Beyond that, mixed responses were observed with regards to the structure, difficulty and appropriateness (for beginners) of the actual learning content provided by the `muco` application. This aligns with the results of other scholars who reported on similar interest increasing effects with STEAM education in a school teaching [12, 19].

The results of the studies conducted also revealed a (re)structuring of the learning content and a visualisation feature as potential areas for future work. Though the overall reaction on the structure and organisation of the learning content was mixed, dividing and restructuring the learning modules into smaller content parts would likely prove helpful and encouraging in the perceived learning experience. This would be especially true for younger audiences. Likewise, visualisations in the learning contexts can serve as a powerful tool for improving both understanding and memorisation of complex topics. Especially when faced with news concepts, visualisation can strongly accelerate and advance understanding and learning. For these reasons, the potential for visualisation features is subjected to further exploration in future iterations.

Apart from these content and design related issues, on a broader level, specific knowledge transfer potentials between the domains of music (including music theory, analysis and composition) and computing (including computational and algorithmic thinking, and programming) remain subject to more extensive empirical research.

Acknowledgments

The authors wish to thank the Center for Technology & Society (CTS)[8] for the funding received. The CTS is an inter-university research platform focusing on inter- and transdisciplinary development of technology, socio-technical research and inter-university cooperation.

8. REFERENCES

[1] H. K. Taube, "Computation and music," *Sound Musicianship: Understanding the Crafts of Music*, 2013.

[2] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[3] S. Laato, S. Rauti, and E. Sutinen, "The role of music in 21st century education-comparing programming and music composing," in *2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT)*. IEEE, 2020, pp. 269–273.

[4] A. R. Brown, "Algorithms and computation in music education," in *The Oxford Handbook of Algorithmic Music*, 2018.

[5] N. Collins, *Origins of algorithmic thinking in music*, 01 2018, pp. 67–78.

[6] A. Misra, D. Blank, and D. Kumar, "A music context for teaching introductory computing," *ACM SIGCSE Bulletin*, vol. 41, no. 3, pp. 248–252, 2009.

[7] S. Aaron, A. F. Blackwell, and P. Burnard, "The development of sonic pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming," *Journal of Music, Technology & Education*, vol. 9, no. 1, pp. 75–94, 2016.

[8] S. Siva, T. Im, T. McKlin, J. Freeman, and B. Magerko, "Using music to engage students in an introductory undergraduate programming course for non-majors," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 975–980. [Online]. Available: https://doi.org/10.1145/3159450.3159468

[9] V. Barr and C. Stephenson, "Bringing computational thinking to k-12: What is involved and what is the role of the computer science education community?" *ACM Inroads*, vol. 2, no. 1, p. 48–54, Feb. 2011. [Online]. Available: https://doi.org/10.1145/1929887.1929905

[10] F. J. García-Peñalvo, D. Reimann, and C. Maday, *Introducing Coding and Computational Thinking in the Schools: The TACCLE 3 – Coding Project Experience*. Cham: Springer International Publishing, 2018, pp. 213–226. [Online]. Available: https://doi.org/10.1007/978-3-319-93566-9_11

[11] P. Fotaris, T. Mastoras, R. Leinfellner, and Y. Rosunally, "Climbing up the leaderboard: An empirical study of applying gamification techniques to a computer programming class." *Electronic Journal of e-learning*, vol. 14, no. 2, pp. 94–110, 2016. [Online]. Available: https://eric.ed.gov/?id=EJ1101229

[12] F. Kayali, V. Schwarz, N. Luckner, and O. Hödl, "Play it again. digitale musikinstrumente im mintunterricht," *journal für lehrerInnenbildung jlb*, vol. 20, no. 1, pp. 54–66, 2020. [Online]. Available: https://elibrary.utb.de/doi/pdf/10.35468/jlb-01-2020_04

[13] E. Perignat and J. Katz-Buonincontro, "STEAM in practice and research: An integrative literature review," *Thinking Skills and Creativity*, vol. 31, pp. 31–43, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1871187118302190

[14] O. Shatunova, T. Anisimova, F. Sabirova, and O. Kalimullina, "STEAM as an innovative educational technology," *Journal of Social Studies Education Research*, vol. 10, no. 2, pp. 131–144, June 2019.

[8] https://cts.wien

[Online]. Available: https://www.learntechlib.org/p/216582

[15] G. Ozkan and U. U. Topsakal, "Investigating the effectiveness of STEAM education on students' conceptual understanding of force and energy topics," *Research in Science & Technological Education*, vol. 0, no. 0, pp. 1–20, 2020. [Online]. Available: https://doi.org/10.1080/02635143.2020.1769586

[16] M. Shamir, M. Kocherovsky, and C. Chung, "A paradigm for teaching math and computer science concepts in k-12 learning environment by integrating coding, animation, dance, music and art," in *2019 IEEE Integrated STEM Education Conference (ISEC)*, 2019, pp. 62–68.

[17] H. Thuneberg, H. Salmi, and F. Bogner, "How creativity, autonomy and visual reasoning contribute to cognitive learning in a STEAM hands-on inquiry-based math module," *Thinking Skills and Creativity*, vol. 29, pp. 153–160, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1871187118301159

[18] T. McKlin, B. Magerko, T. Lee, D. Wanzer, D. Edwards, and J. Freeman, "Authenticity and personal creativity: How earsketch affects student persistence," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 987–992. [Online]. Available: https://doi.org/10.1145/3159450.3159523

[19] A. Xambó Sedó, S. Saue, A. R. Jensenius, R. Støckert, and Ø. Brandtsegg, "NIME prototyping in teams: A participatory approach to teaching physical computing," in *Proceedings of the International Conference on New Interfaces for Musical Expression*. Universidade Federal do Rio Grande do Sul, 2019, pp. 216–221.

[20] D. Herro, C. Quigley, and H. Cian, "The challenges of STEAM instruction: Lessons from the field," *Action in Teacher Education*, vol. 41, no. 2, pp. 172–190, 2019. [Online]. Available: https://doi.org/10.1080/01626620.2018.1551159

[21] P. J. Denning, "The profession of it beyond computational thinking," *Commun. ACM*, vol. 52, no. 6, p. 28–30, Jun. 2009. [Online]. Available: https://doi-org.uaccess.univie.ac.at/10.1145/1516046.1516054

[22] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2012/08/Jeannette_Wing.pdf

[23] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review*, vol. 22, pp. 142–158, 2017.

[24] T.-C. Hsu, S.-C. Chang, and Y.-T. Hung, "How to learn and how to teach computational thinking: Suggestions based on a review of the literature," *Computers & Education*, vol. 126, pp. 296–310, 2018. [Online]. Available: https://doi.org/10.1016/j.compedu.2018.07.004

[25] CSforALL, "CS for ALL computer science for all students," https://www.csforall.org/, 2016, accessed: 2021-07-30.

[26] Computer Science Teachers Association, "CSTA computer science teachers association," https://www.csteachers.org/, 2019, accessed: 2021-07-30.

[27] S. Kumar Basak, M. Wotto, and P. Belanger, "E-learning, m-learning and d-learning: Conceptual definition and comparative analysis," *E-Learning and Digital Media*, vol. 15, no. 4, pp. 191–216, 2018.

[28] K. M. Kapp, *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons, 2012.

[29] K. Squire and H. Jenkins, "Harnessing the power of games in education," *Insight*, vol. 3, no. 1, pp. 5–33, 2003.

[30] P. Mishra, C. Fahnoe, D. Henriksen, D.-P. R. Group *et al.*, "Creativity, self-directed learning and the architecture of technology rich environments," *TechTrends*, vol. 57, no. 1, pp. 10–13, 2013.

[31] J. Lazar, J. H. Feng, and H. Hochheiser, *Research methods in human-computer interaction*, 2nd ed. Morgan Kaufmann, 2017.

[32] J. Sauro and J. R. Lewis, *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016. [Online]. Available: https://www.perlego.com/book/1809426/quantifying-the-user-experience-pdf

[33] J. Nielsen, "Ten usability heuristics," 2005.

[34] B. Shneiderman, C. Plaisant, M. S. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, *Designing the user interface: strategies for effective human-computer interaction*. Pearson, 2016.

[35] A. Blackwell, A. McLean, J. Noble, and J. Rohrhuber, "Collaboration and learning through live coding (Dagstuhl Seminar 13382)," *Dagstuhl Reports*, vol. 3, no. 9, pp. 130–168, 2014. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2014/4420

[36] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *International Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008. [Online]. Available: https://doi.org/10.1080/10447310802205776

Improving Chord Prediction in Jazz Music using Melody Information

Leonhard Driever[*1] leonard.driever@epfl.ch
Mels Loe Jagt[*1] mels.jagt@epfl.ch
Kuan Lon Vu[*1] kuan.vu@epfl.ch
Daniel Harasim[1] daniel.harasim@epfl.ch
Andrew McLeod[1] andrew.mcleod@epfl.ch
Martin Rohrmeier[1] martin.rohrmeier@epfl.ch

[1]École Polytechnique Fédérale de Lausanne (EPFL)
Digital and Cognitive Musicology Lab
[*]Authors 1, 2, and 3 contributed equally to this work.

ABSTRACT

Chord sequence prediction is a well-studied task in music information retrieval that involves predicting the next chord given the sequence of previous chords. It is of practical interest as a part of models for tasks such as chord transcription and automatic accompaniment. Less studied is the interaction between the chord sequence and additional information encoded in the melody. This study investigates whether a long short-term memory network (LSTM) that jointly considers the previous chords and melody notes can achieve higher chord-prediction accuracy than a baseline LSTM that uses only chord progressions as input. The models are trained and evaluated on the Weimar Jazz Database, which comprises transcriptions of melody improvisations over 456 Jazz standards. The results show that the inclusion of melody information significantly improves the chord-prediction accuracy, particularly for rare chords. Qualitative and quantitative investigations of the models' predictions suggest that our model benefits most from melody information in two cases. First, the melody model performs much better in the absence of melody, suggesting that this lack of melody is itself informative. Second, after melody notes that imply rare chords (e.g., notes that occur in diminished or half-diminished chords), the model's performance on those rare chords improves.

1. INTRODUCTION

Harmony plays an important role in a wide variety of styles and genres of music, and the prediction of the next chord in a sequence of chords constitutes a fundamental task in Music Information Retrieval (MIR) [1]. Musicologically, sequential chord prediction is important for the study of the syntax of tonal harmony [2, 3], in particular for advancing the understanding of the structure of chord progressions. From a practical point of view, models for sequential chord prediction are important components in systems performing a variety of tasks, such as chord estimation (from both audio and MIDI) [4–7], harmonic structure detection [8–10], automatic accompaniment [11, 12], and music generation [13,14]. Typically, such systems also have other components such as a segmentation or duration model which predict the locations of chord changes [5, 8]. In this work, however, we use the ground-truth chord segmentation, and instead concentrate purely on the prediction of the next chord in a sequence of chord symbols.

One important axis along which chord sequence prediction models can be categorized is by the chosen model architecture. Skip-gram models, as well as Markovian approaches such as n-gram models, consider a few chords at a time as symbols themselves [15–19]. They typically treat different chords independently rather than using any feature extraction. Multiple viewpoint approaches can account for musical dimensions such as rhythm and pitch separately [19–21]. Non-sequential models, such as generative grammars [22–25], can also be used for sequential chord prediction [26]. While such grammars are strongly rooted in music theory, their practical application typically involves more computational complexity than the aforementioned methods.

Deep learning methods, in particular recurrent neural networks (RNNs), have been the dominant method in recent years due to their flexibility, short development time, and high performance (provided that sufficient training data is avaliable). RNNs are able (in theory) to consider the entire history of chords during prediction [5, 8–10]. They can treat each chord independently (like n-gram models) by using one-hot vectors as input, or can account for multiple musical dimensions simultaneously (like the multiple viewpoint approaches) by using hand-crafted features or multi-hot vectors as input. Additionally, no matter the specific input format and features, chord embeddings can easily be trained (e.g., by Word2Vec [27]) and used in order to learn relationships between similar chords [28, 29].

In this work, we explicitly investigate whether adding information about the past melody (i.e., the melody during previous chords) improves prediction accuracy for the subsequent chords. It has been shown that external, non-chordal information can improve harmonic structure prediction performance, for instance rhythmic and metrical

Copyright: © 2022 Leonhard Driever et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Root note	∈ ({A-G} × {b, #, Ø})
Triadic form	∈ {maj, min, dim, sus4, aug}
Extension (+ alteration)	∈ {7♭/#, 9♭/#, 11♭/#, 13♭/#}
Bass note	∈ { \ + ({A-G} × {b, #, Ø})}

Figure 1: WJazzD's basic chord notation. Every chord consists of at least a root note and a triadic form. Extensions (with optional alteration) and bass note are optional.

information [30]. While it is essential to use *concurrent* melodic information (encoded in the form of chroma vectors [5] or MIDI notes [8]) for automatic chord estimation and harmonization, in this work only the melody prior to the chord to be predicted is taken into account.

It is important to emphasize at this point that we do not intend to create a state-of-the-art system for chord prediction, and therefore a comparison with such methods is outside of the scope of this work. Rather, we present here a controlled experiment investigating the effect of a single feature (past melody information) on chord prediction performance. As a consequence, our findings can inform the design of chord prediction systems in real-time scenarios such as live transcription or automatic accompaniment. However, we make here two simplifying assumptions that are not fulfilled for automatic accompaniment: our model takes as input the ground truth chord segmentation boundaries, and the model gets as input the correct chords from the previous steps. These assumptions are essential to eliminating noise from our experiments and enable direct conclusions about the effect of melody on chord sequence prediction.

A motivating idea for this study is that the melody might contain information about the tonal context of a section of music, which leads to more accurate predictions. Consider the scenario of predicting the chord following a G7. If the melody played over the G7 contains the notes E♭ and A♭, then a C minor chord should be more probable than a C major chord, while the notes E and A would more strongly imply a C major chord. The G7 chord on its own would not disambiguate between these two cases. All code for the models discussed in this work are available online. [1]

2. DATA

2.1 Dataset

In this study, we use the *Weimar Jazz Database* (WJazzD) [31], containing 456 unique jazz solo improvisations. Each solo improvisation contains a transcribed improvised solo melody and the chord accompaniment. Of the tables provided in the WJazzD, only *melody* and *beats* are selected and utilized for further analysis, providing information on the pitches in the melody and the accompanying chord progression respectively for all solos.

In the WJazzD, a chord is defined by a root note (pitch class), a triadic form, extension (and alteration), and the bass note (see Figure 1). The chord extension is any ascending combination of 7, 9, 11 and 13, each accompanied by an optional alteration (flat or sharp). The bass note indicates the lowest note in the chord and is assumed to be the same as the root note if not included. We treat all enharmonically equivalent pitch classes as identical (i.e., Ab and G# are the same).

[1] https://github.com/ldriever/ML_Jazz

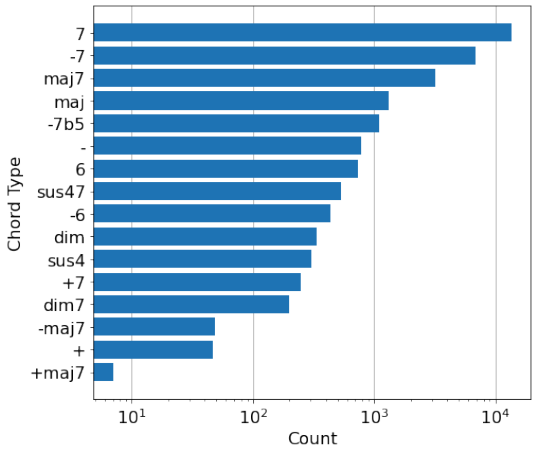


Figure 2: Number of chords per type in the full WJazzD after type reduction.

Each jazz solo is referenced by a unique ***melid*** $\in [1, 456]$. Together with a ***bar*** $\in [-1, N_{\textbf{\textit{melid}}}]$ and ***beat*** $\in [1, 4]$, a unique coordinate to identify a distinct beat in a particular solo is formed. Here $N_{\textbf{\textit{melid}}}$ indicates the max number for ***bar*** specific to a solo. Chords in the WJazzD only change on beats. Thus, utilizing this coordinate system, for each beat in a solo, its corresponding chord can be selected. Each beat also has a duration, measured in seconds. The WJazzD is not quantized, so consecutive beats typically have slightly differing durations.

Notes from the solo improvisations are also referenced by this system, which we use to assign each note to a particular chord, as well as by more fine-grained sub-beat and tatum coordinates, which we use only for ordering the notes. Each note has a MIDI pitch in the range $0 - 127$ and a duration (unquantized), measured in seconds.

2.2 Reduction of Chords

There are 418 unique chord symbols in the dataset, excluding 'NC' which stands for no chord, and which we handle by continuing the previous chord. Some of these chords are very rare in the dataset. To reduce this sparsity, we alter the chord notations by removing chord extensions beyond the 7th (e.g., Ab+79b → Ab+7; D#7913 → D#7). This reduction maps rare chords to simplified and more frequent representations while preserving their core. After this reduction, 297 unique chords remain. By further removing the bass notes for each chord (e.g., C7/A → C7; A+/C# → A+), the number of unique chord types (triadic form plus remaining extensions) is reduced to 16, and the number of unique chords (including their root) is 183. Figure 2 shows the resulting distribution of chord types, which is still highly skewed, but to a much lesser degree than the

original dataset. Following this reduction, 4% of all the chords are identical to the previous chord.

3. MODEL

3.1 Input Representation

In this work, we compare two different model input representations. First, a baseline model, for which the input represents only a sequence of chords, and each input vector represents a single chord. The second model investigates the influence of including information about the improvised solo melody by instead encoding each chord as a *sequence* of input vectors, each representing the chord as well as a single note played during that chord (if there are any).

3.1.1 Baseline

For the baseline model, each tune is represented as a sequence of input vectors, each representing a chord without any information about the improvised solo melody.

Each chord is encoded into a multi-hot vector of length 24. The first 12 entries indicate the notes in a chord. We use a multi-hot chroma vector of length 12 with the root of the chord at 0. Thus, each chord of the same type (e.g., maj7) will be identical in their first 12 elements. The second 12 elements indicate the root note of the chord represented by a one-hot pitch class vector with C=0. By encoding chords in this way, we embed them in a space where some information can be shared across related chord types irrespective of their root (e.g., major chords and augmented chords share a major 3rd above the root) as well as across unrelated chord types with the same root (e.g., C7 and Cdim7 share a root note).

3.1.2 Encoding Melody Information

For the melody model, each input vector represents a single note and its associated chord. If multiple notes occur during a single chord, they are encoded in separate vectors with the same chord representation. Here, each input vector is composed of three concatenated parts, of length 45 in total: the chord (identical to that described for the baseline), the pitch, and the duration.

The pitch of a note encoded in a multi-hot vector of length 20. The first 12 elements are a one-hot vector indicating the enharmonic pitch class of the note, where C=0. The last 8 elements encode the octave of the note from 0–7, where we define middle C to be in octave 3. To complete the note's encoding, the last element indicates the duration of the note, measured as a proportion of the corresponding beat. Since notes in the WJazzD are not quantized, this duration is typically not an even multiple or divisor of the beat, but rather include some noise, indicating a human bias in the length of the note played. In the absence of any notes during a chord, the pitch and duration parts of the input vector are all set to 0.

Since we consider each melody as a sequence of notes (represented by their pitches and durations), but the task is to predict the next chord, the melody model's prediction is only measured after the last note of a given chord. Structurally, this leads to one important difference between the baseline and melody models: The melody model may receive each chord as input multiple times, whereas the baseline model only sees each chord once. To control for this difference, we considered duplicating the baseline's input similarly. However, that would in fact still encode melody information into the baseline model—specifically, the number of notes that occur during each chord, which, as we show in Section 4, is a key piece of information that the melody model relies on for its increased performance.

3.2 Model Architecture

Both baseline and melody models have an essentially identical structure based on a recurrent neural network (RNN). Unlike dense or convolutional neural networks, RNNs are able to retain sequential information and can thus be applied to data, such as music, of variable-length sequences where the order of data points is of key importance. The selected type of RNN is a long short-term memory (LSTM) network [32]. An LSTM makes use of two hidden cell states and an arrangement of three types of "gates" to allow it to retain information over longer sequences of input data. This allows LSTM networks to overcome the "vanishing gradient" problem faced by traditional RNNs [33]. We use uni-directional LSTMs because a bi-directional LSTM would require using information from the entire known chord series, which is not compatible with the aim of chord sequence prediction.

The model architecture is depicted in Figure 4, where "v: l" indicates that the data at that point is a vector of length l. It consists of a single dense embedding layer, one or more LSTM layers (set via hyperparameter tuning; see subsection 4.1), and a single dense output layer. The embedding layer finds a suitable representation of the input data and converts each data point to a representation of size "embedding size" (set by hyperparameter tuning). The LSTM layers handle the sequential information of the data and output data points of size "LSTM hidden size" (set by hyperparameter tuning). The final linear dense neural network layer translates the data into the desired representation: a vector of length 183 where the location of the maximum value indicates the predicted chord. Remember that in the case of the melody model, only the output resulting from the last input note of each chord is used.

4. RESULTS AND DISCUSSION

4.1 Hyperparameter Tuning

For our experiments, we performed 10-fold leave-one-out cross-validation so that each tune in the WJazzD can be included in a test set. Thus, 10 different versions of each model were trained, each with one fold for testing, one for validation, and the remaining 8 for training. We tuned the models' hyperparameters on a random 0.8, 0.1, 0.1 split (not aligned with any of the folds), and used the values which maximized performance on that validation set throughout testing.

To train each model, we use the Adam optimizer [34], with cross entropy as a loss function, and weight decay to

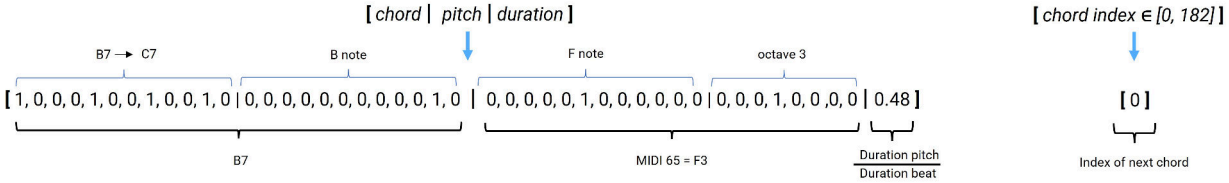


Figure 3: Schematic of input vector (left) and its target (right).

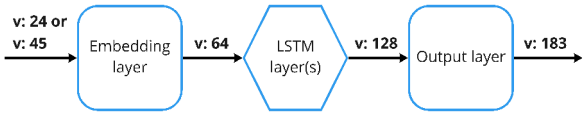


Figure 4: Architecture of the used machine learning model.

implement l_2 regularization. To reduce the amount of overfitting, we apply early stopping to end training when the validation loss has not decreased over the last 30 epochs. To set the batch size and learning rate, we used the automated tuning algorithms of the Pytorch-Lightning module (based in part on [35]) which resulted in a batch size of 64 and a learning rate of 0.014 for each model. Additionally, we optimized the amount of weight decay in the range 10^{-5} to 0.1 for each model, resulting in 0.001 for the baseline and 0.00008 for the melody model.

Finally, we also optimized three structural hyperparameters using a grid search: embedding size $\in \{32, 64, 128\}$, LSTM hidden size $\in \{32, 64, 128, 256\}$, and the number of LSTM layers $\in \{1, 2, 3\}$. The best performing values were 64 and 128 for both models for the embedding size and the LSTM hidden size respectively, and 1 and 2 LSTM layers for the baseline and the melody model respectively. That the melody model is larger was to be expected: It must retain information over longer sequences of input data.

For both models, the training and validation losses remained rather far apart from one another, even at the best identified value of the weight decay. However, this was deemed unavoidable as a sensitivity analysis was performed and showed that any further increases and decreases of the weight decays lead to deteriorated performance in terms of validation accuracy.

4.2 Results

Table 1 reports the average prediction accuracies of the baseline model and the melody model across the entire WJazzD. Note that for the accuracy averaged across each chord, the contributions of different tunes depends on their length. The results show that the additional melody information has lead to an increase in prediction accuracy, from 44.95% to 47.37% when averaged across each chord. For both models, there is a wide spread between maximum and minimum performance when averaged per tune, ranging from $0\% - 100\%$ for the melody model and $0\% - 98.6\%$ for the baseline. Nonetheless, the melody model's increase in performance is consistent across all 10 folds.

To quantify the evidence for the hypothesis that melody information increases prediction accuracy averaged per tune,

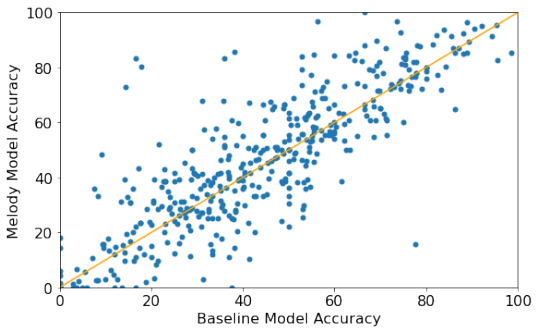


Figure 5: Comparison of prediction accuracy per tune for the baseline and the melody model. The diagonal represents unchanged accuracy.

we performed a Bayesian linear regression that models a piece's accuracy on the basis of the model ID, the tune ID, and the model training ID for each fold (because the chord predictions inside one fold are not independent). We use weakly informative prior distributions over the regression coefficients that weakly favor the alternative hypothesis that melody information does not increase the prediction accuracy. This setup quantifies the uncertainty in how much of the accuracy difference is actually related to the different model architectures by controlling for tune variety and model training (see for example [36, 37] for more information about this statistical testing approach). The regression was implemented via Markov-chain Monte-Carlo sampling using the python package `bambi` [38], which is based on the probabilistic programming language `pymc3` [39]. The results show that the probability that the accuracy differences can be traced back to the model architectures is about 99.52%. In other words, it is over 200 times more likely that melody information increases chord-prediction accuracy than not. This can be considered very strong evidence [40].

Each model's performance per tune is visualized in Figure 5, where points above the diagonal indicate that the melody model improves prediction accuracy compared to the baseline, and points below the diagonal indicate that prediction accuracy decreases for the melody model. Overall, melody information improves prediction accuracy for 49.66% of the tunes in the WJazzD (average accuracy increase 11.94%), had no effect for 12.41% of the tunes, and worsened performance for 37.93% of them (average accuracy decrease 9.48%).

For further investigation, we quantified each model's performance on target chords where melody notes are present

Model	Baseline	With Melody
Average across Chords	44.95	47.37
Standard Error	0.29	0.29
Average across Tunes	43.55	45.88
Standard Error	1.09	1.20

Table 1: Average prediction accuracies in percent for both models across the entire WJazzD.

during the previous chord and those where they are not. Interestingly, the melody model outperforms the baseline by 2.26% when a melody is present, but by 3.80% when it is not. This is a counter-intuitive result: the melody model improves most when there is no melody. However, it appears that the absence of a melody is also a strong signal that the baseline model misses.

A confusion matrix for the baseline model's predictions is shown in Figure 6a, and the melody model's confusion matrix is shown in Figure 6b. Each of these matrices is normalized by row, and "!root" refers to predictions which have an incorrect root and are therefore not included in any other column (the other columns only capture predictions where the root is correct). Chord types are sorted by frequency, and the figures show that both models perform very well on the most common chord type (7), and worse on the more rare chords, which is not unexpected. However, the melody model sees an improvement over the baseline in the more rare chords, especially from maj to dim. This is visualized in Figure 6c, which plots the difference between the two confusion matrices (melody minus baseline model) using a diverging color scale. Here, the improvement of the melody model in correctly predicting chords is shown by red entries along the main diagonal. The blue entries in the "!root" column (the rightmost column in Figure 6) clearly show that the melody model also reduces the proportion of errors due to an incorrect root. In fact, the only chord type for which the baseline model outperforms the melody model is minor triads, which the melody model sometimes misclassifies as rarer 6th and minor 6th chords (although this is not a common mistake, as can be seen in Figure 6b).

To investigate what specific aspects of the melody tend to correlate with the increased performance of the melody model, Figure 7 shows how the presence of each pitch during the chord preceding each target chord influences each model's prediction accuracy. The pitches are denoted by their interval above the root of the target chord. For example, a C7 for which the melody included an A and a B during the previous chord would count towards the "Pitch present" values for M6 and M7 and the "Pitch absent" values for all of the other intervals. The values for the intervals m3, d5, and m6 indicate that the melody model outperforms the baseline to a much greater degree following seeing those pitches than in their absence.

In fact, for these intervals (the three largest increases), the results have a clear music theoretical explanation: after a d5, the proportion of -7b5, dim, and dim7 chords (all of which contain the pitch a d5 above their root) increases to 9.12% versus 5.50% overall. The melody model already

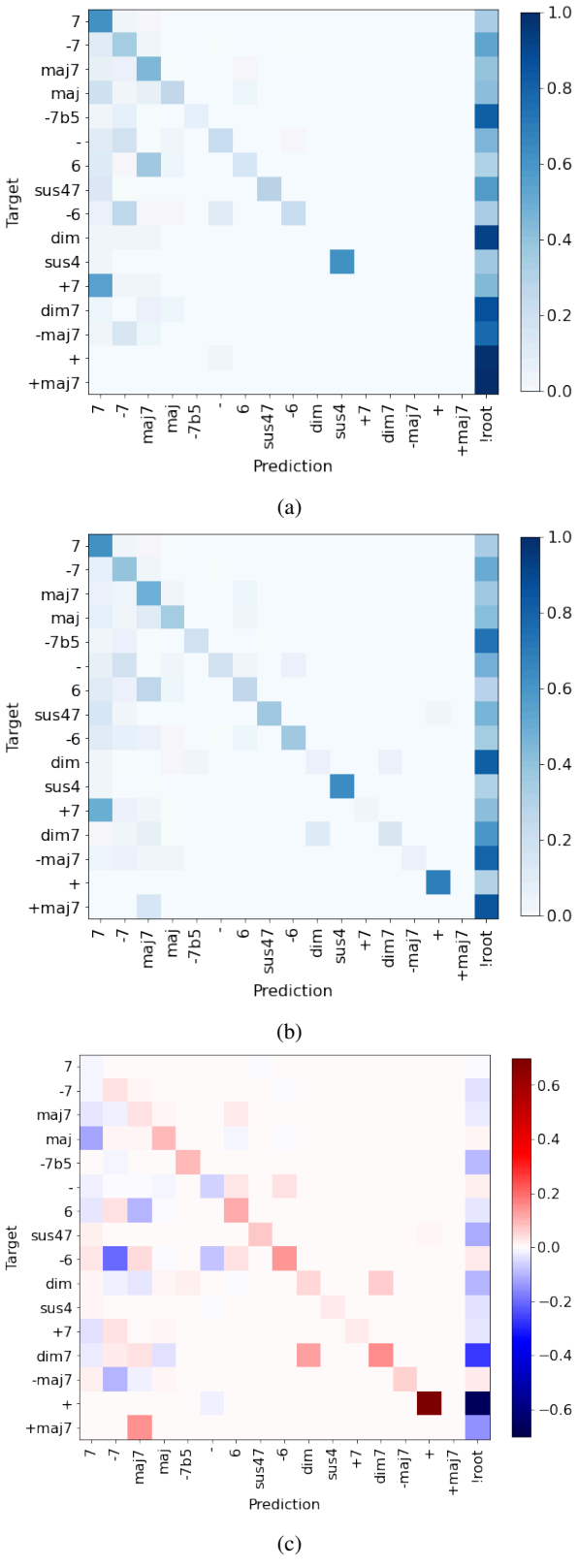


Figure 6: Confusion matrix for the baseline model (a) and the melody model (b), normalized by row. "!root" indicates a prediction with the incorrect root note, which is not included in any other column. The chords are sorted by their frequency in the WJazzD. The difference (b) minus (a), i.e., the melody model's improvement over the baseline, is visualized in (c).

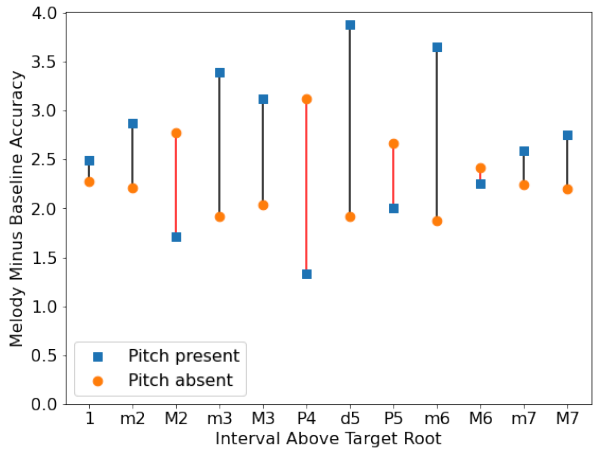


Figure 7: The melody model's improvement over the baseline depending on what pitches (measured by their interval above the target chord's root) co-occur with the chord preceding the target chord (i.e., those which are in the melody model's input).

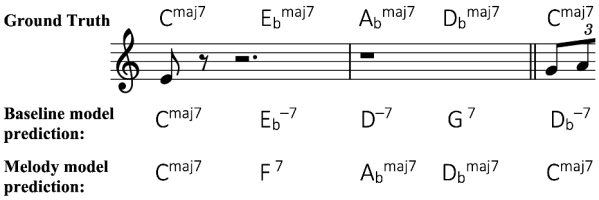


Figure 8: Extract of bars 15–17 from Kenny Dorham's 1955 improvisation over the tune *Lady Bird* [31].

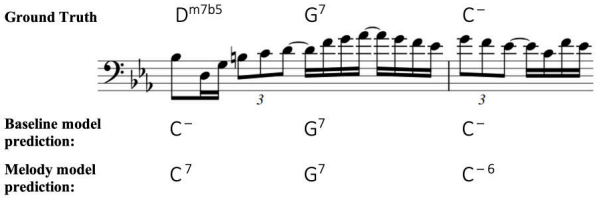



Figure 9: Extract of bars 23–24 from Steve Turre's 1987 improvisation over the tune *Dat Dere* [31].

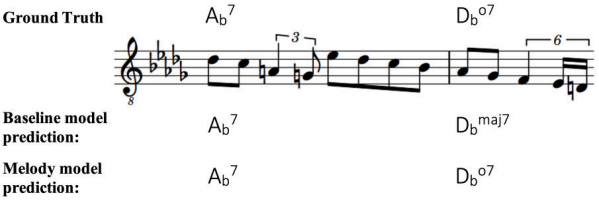



Figure 10: Extract of bars 12–13 from Joe Henderson's 1991 improvisation over the tune *U.M.M.G.* [31]

does better than the baseline on these chords (see Figure 6c), but after a d5, the melody model outperforms the baseline *even more*, by 11.46% compared to 9.32% overall. A similar result is found after m3 and m6 intervals (strong indications of a minor key): the frequency of - and -7 chords is greater after these intervals, and the melody model's increase in performance rises from 3.48% overall to 4.47% in their presence. After a P4, the melody model outperforms the baseline by less than usual on all of the chords rarer than - (and about equals its usual improvement on the more common chords), perhaps because the P4 is both quite common, and not a strong indicator of any of the rare chords music theoretically. The melody model still outperforms the baseline by about 1.3, even in this case.

4.3 Examples

Here, we present four examples of the melody's effect on chord predictions. First, Kenny Dorham's 1955 improvisation over the tune *Lady Bird* [31], for which melody information improves overall performance significantly. The baseline model achieves an accuracy of 27.28% on this tune, while the melody model correctly predicts 45.45% of the chords. This tune is a clear case in which the lack of melody has improved model performance. Bars 15 and 16 (see Figure 8) contain the turnaround *Abmaj7* → *Dbmaj7* → *Cmaj7*. No melody is present for the prediction of these chords. However, as is shown in Figure 8, the melody model correctly predicts each of these chords, while the baseline model misses all of them. This suggests (as we described earlier quantitatively) that the melody model also has the advantage of being able to encode where no melody is played, and to make use of this additional signal.

One example for which melody information has decreased overall performance is Joshua Redman's 1994 solo over the tune *Sweet Sorrow* [31]. For this tune, the baseline model achieves an accuracy of 28.57% while the melody model achieves only 11.43%. Inspecting the chord progression and sheet music for *Sweet Sorrow*, one finds that for most of the tune, the chord progression constantly repeats the chords *Eb7* → *Ab-6*. The melody, however, changes. The model without melody information settles into a repeating pattern of *Eb7* → *Ab7* and thus predicts half of this main chord pattern correctly. The model with melody information, on the other hand, predicts a more varying chord pattern. Thus, it appears that in this case the combination of a steady chord pattern and varying melody means that including the melody information more noise than useful information.

Looking into specific outputs, Figure 9 shows an excerpt from Steve Turre's improvisation over *Dat Dere*. Here, the melody model predicts C-6 for the final chord, while the baseline correctly predicts C-. As mentioned in regards to Figure 6, this is one example where the melody model overpredicts a more complicated chord, in particular the - → -6 mistake. Furthermore, related to Figure 7, this example shows a case in which a P4 in the melody (the two Fs on beats 3 and 4 of the first bar, which are a fourth above the target C), do not imply any chord in particular.

Finally, Figure 10 presents an excerpt from Joe Henderson's improvisation over *U.M.M.G.*. Here, the melody model correctly predicts the final Dbdim7, while the baseline instead guesses Dbmaj7 (a chord which occurs multiple times previously). Again related to Figure 7, this is one example where a d5 in the melody (the G in the first bar—enharmonic to Abb—which is a diminished 5th above the target Db) appears to have been particularly informative for the model.

5. CONCLUSION

In this work, we presented two simple LSTM models for the task of chord prediction in Jazz, differing only by their input representation and tuned hyperparameters. We have shown that including improvised melody information in the input data leads to more accurate chord predictions. The main increase in performance is on the more rare chords, and this improvement is even more evident when the melody itself contains a note which implies a rare chord. We also showed that, for the melody model, the absence of melody was itself an informative signal.

Future work could investigate whether a similar finding is true for other genres of music. Since Jazz improvisations can be more unrelated to the underlying chord progression than melody from other styles of music (e.g., Western classical or Pop music), it is an open question as to whether a similar improvement can be found for those styles. Including the specific performer as an additional input feature (the WJazzD contains multiple improvisations by each performer) could also test whether the melody model would be able to perform lick detection or improvisation style detection. The information of the performer would help to differentiate melody cues commonly used by each performer, which could lead to more accurate chord prediction. Finally, integrating melody information into a system which must also perform chord segmentation and does not have access to the correct prior chords would be a logical next step, along with updating the model to handle more complicated polyphonic structures that are important to harmonic structure [41].

Acknowledgments

This project has received partial funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement no. 760081-PMSB. It was also partially funded through the Swiss National Science Foundation (SNF) within the project "Distant Listening – The Development of Harmony over Three Centuries (1700–2000)" (grant no. 182811). The authors thank Claude Latour for supporting this research through the Latour Chair in Digital Musicology at EPFL.

6. REFERENCES

[1] M. Schedl, E. Gómez Gutiérrez, and J. Urbano, "Music information retrieval: Recent developments and applications," *Foundations and Trends in Information Retrieval*, vol. 8, no. 2-3, pp. 127–261, 2014.

[2] M. Rohrmeier and M. Pearce, "Musical Syntax I: Theoretical Perspectives," in *Springer Handbook of Systematic Musicology*, R. Bader, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 473–486.

[3] M. Pearce and M. Rohrmeier, "Musical Syntax II: Empirical Perspectives," in *Springer Handbook of Systematic Musicology*, R. Bader, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 487–505.

[4] H. Papadopoulos and G. Peeters, "Large-scale study of chord estimation algorithms based on chroma representation and hmm," in *2007 International Workshop on Content-Based Multimedia Indexing*. IEEE, 2007, pp. 53–60.

[5] F. Korzeniowski and G. Widmer, "Improved chord recognition by combining duration and harmonic language models," in *International Society for Music Information Retrieval (ISMIR)*, 2018.

[6] Y. Wu, T. Carsault, E. Nakamura, and K. Yoshii, "Semi-supervised neural chord estimation based on a variational autoencoder with latent chord labels and features," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2956–2966, 2020.

[7] D. Temperley, "A unified probabilistic model for polyphonic music analysis," *Journal of New Music Research*, vol. 38, no. 1, pp. 3–18, mar 2009.

[8] A. McLeod and M. Rohrmeier, "A modular system for the harmonic analysis of musical scores using a large vocabulary," in *International Society for Music Information Retrieval (ISMIR)*, 2021, pp. 435–442.

[9] G. Micchi, K. Kosta, G. Medeot, and P. Chanquion, "A deep learning method for enforcing coherence in automatic chord recognition," in *International Society for Music Information Retrieval (ISMIR)*, 2021, pp. 443–451.

[10] N. Nápoles López, M. Gotham, and I. Fujinaga, "Augmentednet: a roman numeral analysis network with synthetic training examples and additional tonal tasks," in *International Society for Music Information Retrieval (ISMIR)*, 2021, pp. 404–411.

[11] I. Simon, D. Morris, and S. Basu, "Mysong: automatic accompaniment generation for vocal melodies," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2008, pp. 725–734.

[12] T. Carsault, A. McLeod, P. Esling, J. Nika, E. Nakamura, and K. Yoshii, "Multi-step chord sequence prediction based on aggregated multi-scale encoder-decoder networks," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2019, pp. 1–6.

[13] G. Brunner, Y. Wang, R. Wattenhofer, and J. Wiesendanger, "Jambot: Music theory aware chord based generation of polyphonic music with lstms," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2017, pp. 519–526.

[14] D. Conklin, "Chord sequence generation with semiotic patterns," *Journal of Mathematics and Music*, vol. 10, no. 2, pp. 92–106, 2016.

[15] F. C. Moss, M. Neuwirth, D. Harasim, and M. Rohrmeier, "Statistical characteristics of tonal harmony: A corpus study of Beethoven's string quartets," *PLOS ONE*, vol. 14, no. 6, Jun. 2019.

[16] T. De Clercq and D. Temperley, "A corpus analysis of rock harmony," *Popular Music*, pp. 47–70, 2011.

[17] C. Finkensiep, M. Neuwirth, and M. Rohrmeier, "Generalized skipgrams for pattern discovery in polyphonic streams," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 547–553.

[18] D. R. Sears, A. Arzt, H. Frostel, R. Sonnleitner, and G. Widmer, "Modeling harmony with skip-grams," in *18th International Society for Music Information Retrieval Conference*, 2017.

[19] M. Rohrmeier and T. Graepel, "Comparing feature-based models of harmony," in *Proceedings of the 9th International Symposium on Computer Music Modelling and Retrieval*, 2012, pp. 357–370.

[20] D. Conklin and I. H. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995.

[21] M. T. Pearce, "Statistical learning and probabilistic prediction in music cognition: mechanisms of stylistic enculturation," *Annals of the New York Academy of Sciences*, vol. 1423, no. 1, pp. 378–395, Jul. 2018.

[22] M. Rohrmeier, "The syntax of jazz harmony: Diatonic tonality, phrase structure, and form," *Music Theory and Analysis*, vol. 7, no. 1, pp. 1–63, 2020.

[23] M. Granroth-Wilding and M. Steedman, "A Robust Parser-Interpreter for Jazz Chord Sequences," *Journal of New Music Research*, vol. 43, no. 4, pp. 355–374, Oct. 2014.

[24] D. Harasim, "The Learnability of the Grammar of Jazz: Bayesian Inference of Hierarchical Structures in Harmony," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, Switzerland, 2020.

[25] D. Harasim, M. Rohrmeier, and T. J. O'Donnell, "A Generalized Parsing Framework for Generative Models of Harmonic Syntax," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018, pp. 152–159.

[26] S. A. Herff, C. Finkensiep, and M. Rohrmeier, "Hierarchical syntactic structure predicts listeners' sequence completion in music," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 43, Vienna, 2021, pp. 903–909.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, vol. 26, 2013, pp. 3111–3119.

[28] E. Anzuoni, S. Ayhan, F. Dutto, A. McLeod, F. C. Moss, and M. Rohrmeier, "A historical analysis of harmonic progressions using chord embeddings," in *Sound and Music Computing Conference (SMC)*, 2021, pp. 284–291.

[29] S. Madjiheurem, L. Qu, and C. Walder, "Chord2vec: Learning musical chord embeddings," in *Proceedings of the constructive machine learning workshop at 30th conference on neural information processing systems*, 2016.

[30] D. Harasim, T. J. O'Donnell, and M. Rohrmeier, "Harmonic Syntax in Time: Rhythm Improves Grammatical Models of Harmony," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, Delft, 2019.

[31] M. Pfleiderer, K. Frieler, J. Abeßer, W.-G. Zaddach, and B. Burkhart, Eds., *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017.

[32] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, 1997.

[33] P. Le and W. Zuidema, "Quantifying the vanishing gradient and long distance dependency problem in recursive neural networks and recursive LSTMs," in *Proceedings of the 1st Workshop on Representation Learning for NLP*, 2016.

[34] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[35] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2017, pp. 464–472.

[36] D. J. Barr, R. Levy, C. Scheepers, and H. J. Tily, "Random effects structure for confirmatory hypothesis testing: Keep it maximal," *Journal of Memory and Language*, vol. 68, no. 3, pp. 255–278, 2013.

[37] A. J. Milne and S. A. Herff, "The perceptual relevance of balance, evenness, and entropy in musical rhythms," *Cognition*, vol. 203, p. 104233, 2020.

[38] T. Capretto, C. Piho, R. Kumar, J. Westfall, T. Yarkoni, and O. A. Martin, "Bambi: A simple interface for fitting bayesian linear models in python," 2020.

[39] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, "Probabilistic programming in python using pymc3," *PeerJ Computer Science*, vol. 2, p. e55, 2016.

[40] R. E. Kass and A. E. Raftery, "Bayes factors," *Journal of the american statistical association*, vol. 90, no. 430, pp. 773–795, 1995.

[41] L. Wall, R. Lieck, M. Neuwirth, and M. Rohrmeier, "The Impact of Voice Leading and Harmony on Musical Expectancy," *Scientific Reports*, vol. 10, no. 1, pp. 1–8, 2020.

Towards Lightweight Architectures for Embedded Machine Learning in Musical Instruments

Chris Kiefer
Experimental Music Technologies Lab, University of Sussex
c.kiefer@sussex.ac.uk

ABSTRACT

It can be challenging to engage with machine learning in restricted computing environments, such as the systems often in use in digital or hybrid musical instruments. We often use low power, low memory devices with limited computational power, and need high frequency models for sensor and sound processing. Conversely, contemporary machine learning and AI can be resource hungry, limiting its use in embedded systems. In previous research, the Stochastic Logic Optimisation algorithm offered a method of lightweight machine learning using two-state logic networks, intended for musical use in embedded systems. This experiment shows how this approach can be expanded on, using random boolean reservoirs, for signal generation. These initial results demonstrate the efficacy of a reservoir computing approach, built only from networks of lookup tables. They show that, for the task of training sine wave generators, reservoirs can be improved if built with hierarchical growth algorithms, and further improved by selecting inputs and outputs based on network centrality. The results also demonstrate successful use of Pulse Density Modulation for signal encoding.

1. INTRODUCTION

There are many reasons for which we may want to embed machine learning (ML) processes within a physical musical instrument. Embedding, in this context, will refer to ML running on computing resources within a self-contained instrument, maybe connected to sensors, actuators, and/or input/output ports. Examples could be eurorack modules, hybrid double basses or gestural controllers, among many other possibilities. These instruments may be defined as *smart* [1] or *intelligent* [2]. ML could be used for pattern recognition, sound generation, signal processing, again among many possibilities. We might embed ML for pragmatic reasons; it could be advantageous for an instrument to be portable, to take it to rehearsals and performances, expanding creative opportunity. It may be important to capture and process data quickly within an instrument, with minimum delay. For example, in a feedback instrument such as the halldorophone [3], delays in signal processing would affect the feedback loop and therefore change the sound and behaviour of the instrument. Beyond pragmatic reasons, one can conceptualise machine learning in music and instrument design holistically; that is to consider all the typical processes in ML, data collection, training, inference, experimentation with architectures, as creative materials, as part of a broader musical engagement with ML. If this is the case, then surely our instruments are good interfaces for engaging with ML, beyond conventional interaction paradigms, and with embedded ML we might explore the value of seamless engagement with self-contained ML materials.

Our current batch of ML technologies are typically *heavyweight*; we have large models that can take hours, days or weeks to train, consuming large amounts of energy, memory and computing resources. The implications for musical instrument designer are that, to use these technologies, an instrument needs to be tethered to a powerful computer. This approach was several potential disadvantages; the interface to the instrument is spread across different interfaces and interaction styles, the instrument lacks portability, the connections can introduce latency, Some large models may not run in realtime, although projects such as RAVE [4] are focusing on this issue. To embed ML within an instrument, we need to use smaller computers or microcontollers, that typically have restricted memory and processing resources. We may also want to run low-power devices, perhaps so we can use battery power for portability, or perhaps because it's challenging to try and quietly cool down powerful processing units that would dissipate a great deal of heat within a musical instrument.

Fortunately, low-powered machine learning (or TinyML) is a highly active research field. Outside of computer music applications, one can find successful models such as Bonsai [5], which run successfully in low memory environments without the need for hardware floating point processors. Models have also been configured to run on Field Programmable Gate Arrays (FPGAs) (e.g LogicNets [6], LUTnets [7]), which offer further performance advantages.

Within the field of computer music, research into embedded ML within musical instruments is an emerging field. Possibly because of the challenging demands that musical instruments might place on ML models, embedding ML has been highly technically challenging. However, with increasingly powerful embedded computers (e.g. Teensy 4, EPSP32) and specialist embedded audio platforms (e.g. Bela, Daisy), and as FPGAs become more affordable with open source toolchains (e.g. Lattice ICE40), we have better hardware options to explore this field. As this hardware

Copyright: © 2022 Chris Kiefer et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

collides with TinyML research, we are finding better technologies to experiment with in musical instrument design.

2. STOCHASTIC LOGIC OPTIMISATION

One response to the challenge of embedded musical ML is Stochastic Logic Optimisation (SLO). Where TinyML typically takes trained models and reduces and approximates them until they will run on a more restricted computer, SLO attempts to build fast models from the ground upwards, with basic computing resources. It's an algorithm for optimising feed-forward networks of lookup tables (LUTs), which are the raw material of typical FPGAs and easily translated to run on microcontrollers. A lookup table is a logic block, usually with 4 or 6 inputs, a single output, and a block of data containing an output value for each possible combination of inputs. For example, a 4-input LUT will have a memory containing 16 binary output values, indexed by the value of the input.

LUT networks have the advantage of needing little memory, and running without the need for potentially expensive numerical processing, such as floating point arithmetic. SLO is described in detail in [8]. In brief, the algorithm is a supervised learning method; it takes a randomly initialised network of LUTs and adjusts their truth tables such that the network models the relationship between inputs and outputs in the training data set. It does this by presenting the data in batches to the network, collecting potential truth table adjustments using the heuristic of finding the most minimal adjustment to correct each error, and then applying the most popular of these adjustments.

Initial experiments with SLO focused on a simple beat prediction task. Having established the basic efficacy of the algorithm with this model, the next stage in development towards meaningful use of embedded in musical instruments is to explore more complex signal processing. In the next section, new architectural options for supporting this task are explored, before an experiment to explore their value is reported on.

3. NEW MODEL ARCHITECTURE

3.1 Modelling Time

Representation of time is key to most machine learning models that will be used within musical instruments. Broadly speaking, time in typical machine learning architectures might be represented in one of three ways:

1. Stateless models, with time represented in the data. For example, a feed-forward network of LUTs, as mentioned above, has no memory, so time must be modelled in the training data. Each frame of data might contain a history of the most recent values in the input signal.

2. Stateless models, with feedback from output to input. With a feedback connection, the stateless model becomes a kernel in a dynamical system, giving the model a time-varying reaction to new input.

3. Stateful models, possibly also with output-input feedback connections. These models have internal recurrent connections, and therefore retain their own memory of previous input and internal states.

In the context of this project, stateful models seem most appealing for realtime signal processing. Time and function are united in the model, meaning that any adjustments to the model will be coherent across both domains. This could make training conceptually more simple. This coherence might improve the models potential for realtime manipulation (for example, manipulation of CCRNNs [9]) or network bending approaches [10].

3.1.1 Random Boolean Networks as Reservoirs

In the search for a realtime approach to recurrent machine learning models, reservoir computing [11] offers potentially valuable solutions. In the reservoir computing paradigm, a randomly constructed recurrent model (the reservoir) is paired with a stateless feed-forward model (the readout). In training, observations are collected of how the reservoir reacts to input over time, and the readout is optimised to make predictions based on these observations. The reservoir essentially provides time-varying state to a stateless model, and also functions to map signals into a higher dimensional space which can offer new perspectives on the input data.

Reservoirs can take many forms; a typical configuration is the one used in Echo State Networks [12], with continuously valued neurons updated in discrete time steps. Previous research has also explored boolean reservoirs, in the form of Random Boolean Networks (RBNs). RBNs were introduced by Kauffman [13] as a method for exploring complex dynamics between genes. More recent research has demonstrated the value of RBNs in reservoir computing [14], including successful implementation on FPGAs [15]. Constructed with simple LUTs, RBNs offer a straightforward and lightweight method for reservoir construction using basic computing resources. Previous research has combined RBNs with conventional readout layers, which might run too slowly on embedded systems due to the need for floating point processing. A model that combines an RBN with a trainable LUT network as a readout layer could offer a lightweight architecture for realtime signal processing. For simplicity, this combined model can be called a SLORBN. Figure 1 illustrates a small-scale example of this architecture.

3.1.2 Reservoir Design

An RBN is a network of LUTs. Typically, when used as reservoirs, RBNs are connected together with an architecture determined by linear probability; these *monolithic* reservoirs have no globally imposed structure. Xuan et. al [16] propose an approach to structuring networks to resemble the structure of real-world networks. Their growth algorithm builds networks with hierarchically connected modules. This algorithm has been investigated in the design of RBN reservoirs [17], showing that *modular* reservoirs outperform monolithic reservoirs in temporal tasks: pattern detection, food foraging, and memory recall. These

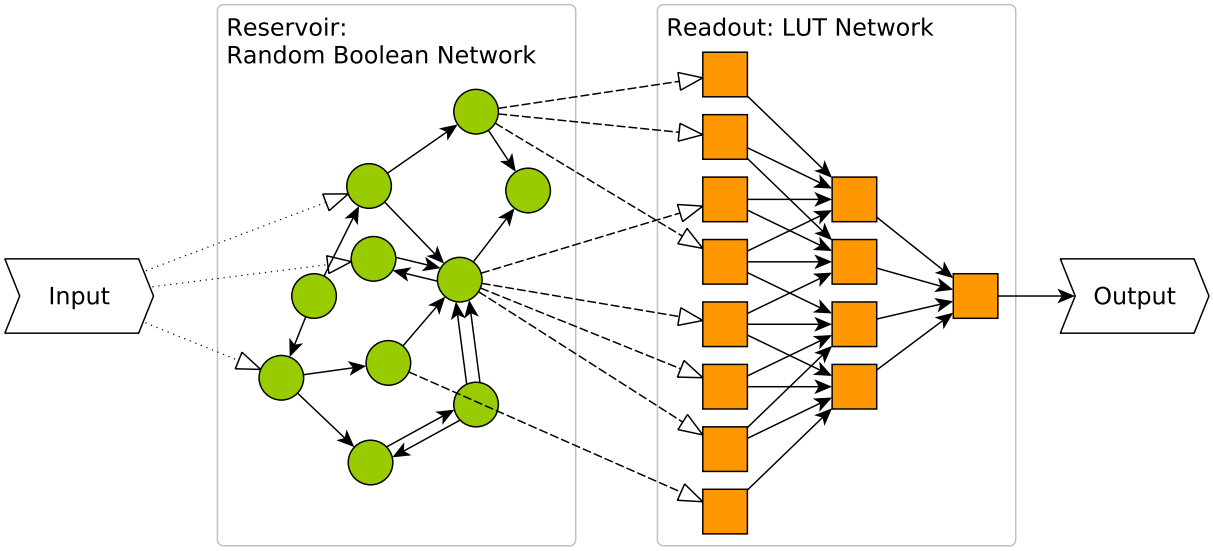


Figure 1: An example of a SLORBN model. The entire model is constructed from LUTs. The input and readout layers are connected to subsets of RBN nodes.

results show promise for further exploration of signal processing tasks that might be useful in musical instrument design. The difference between modular and monolithic RBNs is illustrated in Figure 2. It shows two examples of connectivity in randomly generated 500 node RBNs, the modular RBN exhibiting visible structure across small communities of nodes.

An unexplored extension to reservoir architecture design is the selection of points of connection to inputs and to the readout layer. In examples in the research literature, these are typically chosen with linear probability. Perhaps modular reservoirs, through more structured design, offer optimal connection points compared to random selection?

3.2 Signal Encoding

If we are to use LUT networks as signal processors in musical instruments, RBNs offer a solution to embedded time into the models, but how do we represent continuous signals to processors that work only with binary logic? There are many ways in which numbers can be represented, with standard binary or binary encoded decimal encodings, or using sparse positional encodings such as one-hot encoding. These types of encodings might be challenging because of the demands put onto the model to engage with the encoding abstraction. If binary number encodings are used, the model will have the additional demands of learning to process binary abstractions. Positional encoding will force the models to use large architectures to cope with the scale of inputs and outputs, for example a one-hot encoded 8 bit number would need 256 inputs to connect it to the network.

A third option is the use of Pulse Density Modulation (PDM) [18]. With PDM, a continuous signal is encoded in binary as a series of high-frequency pulses. Figure 3 shows an example, where a sinewave has been encoded using PDM, with 8x oversampling. A continuous signal can be recovered from the PDM signal using a low-pass filter. This system is very commonly used in sigma-delta conversion in DACs and ADCs. In relation to AI and ML, PDM sits within a broader class of *time-based* signal representations that have a neurobiological basis, and are being explored within neuromorphic computing [19] and with spiking neural networks (e.g. [20]). The applications of PDM in computer music and machine learning are under-explored research areas.

Within the SLORBN architecture, the PDM signal is fed into the RBN reservoir and a randomly predetermined subset of the reservoir states are collected. Figure 4 shows an example of these states. The model is then trained with SLO to map from these states to the PDM-encoded target output. During inference, the output of the model is low-pass filtered to recover a continuous signal.

The use of PDM encoding to represent signals in RBNs and LUT networks could be advantageous because it does not impose any extra abstractions on the model, the signal is already using the raw representations that the model is working with. The potential disadvantage is that PDM signals require a higher sample rate than the source signal being encoded, which may also have a follow-on effect of needing larger networks, both of which impose addition demands on computing resources.

4. EXPERIMENT: TRAINING SIGNAL GENERATORS

So far, some key issues have been explored in how trainable LUT networks can be expanded to work as signal processors. A potentially value architecture is the SLORBN: a combination of RBN reservoirs with LUT network readout layers, trained using SLO. The following questions could be considered:

(a) A Monolithic RBN

(b) A Modular RBN

Figure 2: Examples of connectivity in 500 node RBNs, with connections determined by (a) linear probability and (b) Xuan et al's hierarchical growth algorithm

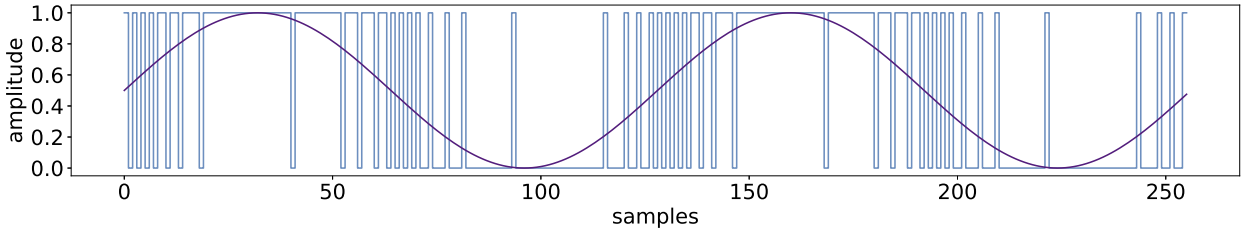


Figure 3: An example of Pulse Density Modulation encoding: a sine wave and the corresponding bit pattern, using 8x oversampling.

1. Does this architecture enable the training of signal processors?

2. How should reservoirs be designed optimally for signal processing? e.g. are monolithic or modular reservoirs better?

3. How should input and output nodes be selected in the reservoir?

4. Is PDM encoding effective for signal processing with these models?

4.1 Method

Inspired by early reservoir computing experiments in trainable sine wave generation [21], an experiment was designed to begin exploration these questions. The task in the experiment was to train a sine wave generator using SLORBN models. This task is a good test of time representations in the model; a sine wave moves in different directions from identical values in each half of the waveform. Without historical context, the model cannot determine which value to produce next.

Full sourcecode for the experiment can be found in the accompanying repository [1] . The original signal was a sine wave, with a 16 sample wavelength. This signal was PDM encoded with 2x oversampling (a higher rate was not necessary because of the lack of high frequency content). To generate a dataset, the PDM signal was repeated 4 times, and then copied into input-output pairs such that the model would be trained to predict the next sample in the signal.

During the experiment, reservoirs were generated in a variety of configurations. They were either monolithic, or were generated using Xuan et. al's hierarchical growth algorithm, with $M = 4$, $n = 4$ and $m_0 = 2$. Inputs and outputs to the reservoir were selected with one of four algorithms, as shown in table 1. The most basic scheme was to choose nodes with linear probability **(RAND)**. It seemed interesting to explore the connectivity within the reservoir and select points based on this information. Reservoirs were analysed using betweenness centrality [22], which essentially measures the importance of each node in a network . Nodes were selected with higher probability for high **(BCH)** or low **(BCL)** betweenness centrality. The final selection technique **(MOD)** applies only to modular

[1] `https://github.com/chriskiefer/SLO_RBN`

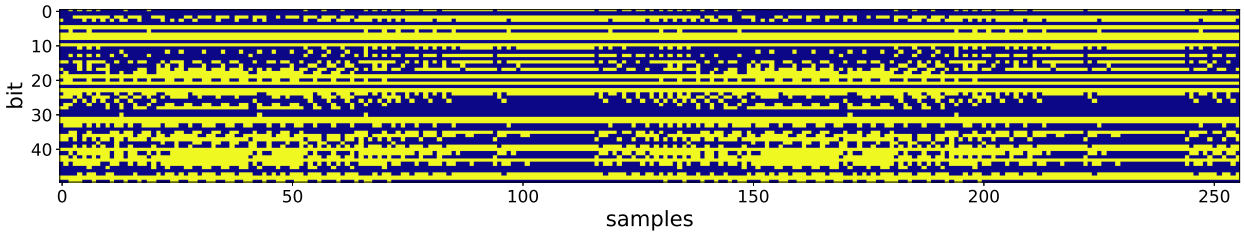


Figure 4: An example of activity within a Random Boolean Network. A 500 node RBN was purturbed by two cycles of a PDM encoded sine wave; this figure shows how the outputs of a subset of these nodes varied over time.

networks; nodes were selected from a minimal set of low-level modules within the hierarchy.

The experiment consisted of multiple trials, where an attempt was made to train a model with this dataset, with randomly selected variations in parameters. Each trial ran as follows:

1. A 500 node reservoir was generated, with a randomly chosen configuration for (a) generation method, (b) input selection and (c) output selection. Table 1 shows these configurations. The numbers of input and output nodes were fixed at 30% for input and 15% for output (manual tuning showed these values to be capable of producing reasonable results). The reservoir generation algorithms were tuned so that the average connectivity in all reservoirs was likely to be close to 2 inputs per node. Reservoirs with this value, referred to as K in RBN literature, exhibit behaviour at the border between order and chaos [23], and are most suited for computation.

2. The input signal was fed into the reservoir, and the corresponding states were collected

3. These states, together with the target output signal, were combined into a dataset, and used to train a LUT network, using SLO. All LUT networks had the following number of nodes per layer: $[4096, 1024, 256, 64, 16, 4, 1]$. Training took place over 600 iterations, with a batch size of 32. Again, these values were selected using manual tuning, such that all configurations had the potential to produce reasonable results.

4. The trained LUT network was evaluated on the original set of reservoir states to obtain an accuracy score, measured as the percentage of correct predictions.

Once successfully trained, a model could be used as a sine wave generator by cueing the reservoir from it's original state, and then feeding the output from the model back into the inputs.

4.2 Results

The experiment took place over a total of 25000 trials, split equally across the configurations. Results are shown in figure 5. The results show that the highest scoring networks were modular, with both inputs and outputs selected based on high betweenness centrality. Monolithic networks with an identical selection method were the second highest scoring. In all comparable cases except BCH/BCL, modular networks had a higher average accuracy than monolithic networks.

Config	Reservoir Type	Inputs	Outputs
0	Monolithic	RAND	RAND
1		BCH	BCH
2		BCL	
3		BCH	BCL
4		BCL	
5	Modular	RAND	RAND
6		BCH	BCH
7			BCL
8			MOD
9		BCL	BCH
10			BCL
11			MOD
12		MOD	BCH
13			BCL
14			MOD

Table 1: Configurations for generating reservoirs in the experiment

5. DISCUSSION AND CONCLUSIONS

This experiment has explored the value of potential expansions of LUT networks and the SLO algorithm, towards applications in embedded ML in musical instruments. It tested a new architecture, where time is represented within a Random Boolean Network reservoir, and the feed-forward readout layer of LUTs is trained with SLO, based on observations from the reservoir.

The experiment shows that this architecture is capable, with high accuracy, of training models as PDM-encoded sine wave generators at low resolution (16 samples, 2x oversampling). The model further explored reservoir generation, input and output connectivity and signal encoding schemes. The results showed that in the specific context of this task, hierarchical modular reservoirs had higher average accuracy than monolithic ones in all but one comparable configuration. This is consistent with results from other comparisons of these network structures. The results on input/output node selection showed that selecting both

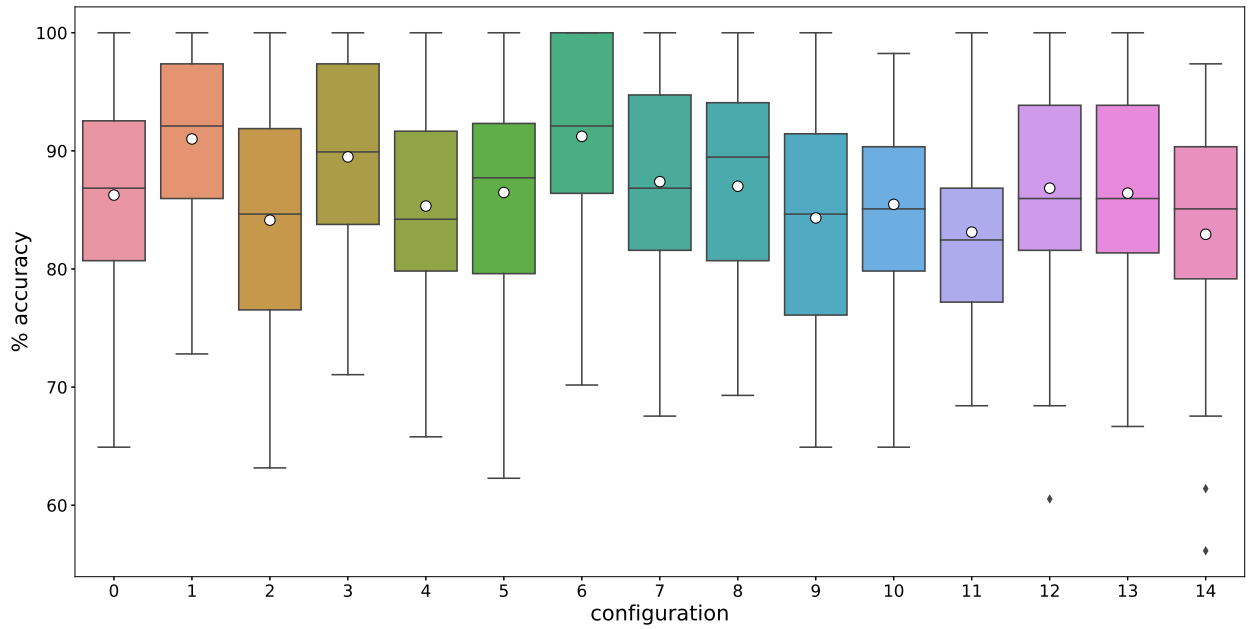


Figure 5: A boxplot of results of the experiment, showing % accuracy for each configuration, as detailed in table 1 The white dots show the averages.

input and output nodes with probability proportional to betweenness centrality gave the best results with both monolithic and modular RBNs.

Pulse Density Modulation encoding has been shown to work well with this architecture, on the basis that high accuracy scores were obtained. This experiment did not compare PDM to other encoding schemes, but it does establish basic efficacy for its use with SLORBN models. The challenge with this encoding is that the model must be oversampled if it needs to respond to or generate higher frequencies. In implementation, this places a burden on microcontrollers because of their sequential processing architecture. FPGAs, in contrast, are ideal for this task, because of the way in which they can be configured with massively parallel LUT circuits. RBNs can be run in a single clock cycle (potentially taking low numbers of nanoseconds) or even unclocked [15] at the maximum speed of the circuitry. Speed in FPGA implementations is independent of scale because of parallelism. Similarly, the readout networks can run at one clock cycle per layer. On both FPGAs and microcontrollers, PDM signals can be output directly to a digital pin, and filtered with relatively simple analogue circuity to recover the underlying signal. PDM processing certainly has interesting potential, and deserves further exploration.

While these results are promising, there is still some way to go towards creating models that work at a resolution high enough for full-frequency audio, while also balancing a minimal architecture for embedded systems. Further investigation showed that the most successful configuration (6) could produce 100% accurate models with a smaller architecture of a 200 node RBN and readout layer sizes of $[256, 65, 16, 4, 1]$, utilising 9.1% or the resources of the models used in the original experiment. Further work should explore the effects of raising resolution in this configuration, to achieve higher oversampling rates and to test higher bandwidth and lower frequency waveforms.

This experiment establishes foundations for an architecture for SLO training and LUT networks that can be used for signal generation. It also raises some interesting questions. While this experiment demonstrates the potential for signal generation, it does not explicitly explore signal processing. For example, how can these models be trained to modulate the signal they are generating? How can they be trained as filters? Another potentially valuable avenue is in network bending. With modular reservoirs, we have plenty of information about their structure, with the addition of information about nodes centrality from network analysis. Perhaps this information can inform network bending approaches, and offer points for manipulation of these models?

Future work will explore hierarchical architectures in trainable LUT networks and to develop further creative applications. It will also focus on finding meaningful benchmarks for these models on various low-power computing devices.

6. REFERENCES

[1] L. Turchet, "Smart musical instruments: vision, design principles, and future directions," *IEEE Access*, vol. 7, pp. 8944–8963, 2018.

[2] T. Magnusson, "Of epistemic tools: Musical instruments as cognitive extensions," *Organised Sound*, vol. 14, no. 2, pp. 168–176, 2009.

[3] H. Úlfarsson, "Feedback mayhem. compositional affordances of the halldorophone discussed by its users,"

in *Proceedings of the 2019 International Computer Music Conference*, 2019.

[4] A. Caillon and P. Esling, "Rave: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[5] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 kb ram for the internet of things," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1935–1944.

[6] Y. Umuroglu, Y. Akhauri, N. J. Fraser, and M. Blott, "Logicnets: Co-designed neural networks and circuits for extreme-throughput applications," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2020, pp. 291–297.

[7] E. Wang, J. J. Davis, P. Y. Cheung, and G. A. Constantinides, "LUTNet: Rethinking inference in FPGA soft logic," in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2019, pp. 26–34.

[8] C. Kiefer, "Stochastic optimisation of lookup table networks, for realtime inference on embedded systems," in *2nd Conference on AI Music Creativity*, 2021.

[9] ——, "Sample-level sound synthesis with recurrent neural networks and conceptors," *PeerJ Computer Science*, vol. 5, p. e205, 2019.

[10] M. Yee-King and L. McCallum, "Studio report: sound synthesis with ddsp and network bending techniques," in *2nd Conference on AI Music Creativity*, 2021.

[11] M. Lukoševičius, H. Jaeger, and B. Schrauwen, "Reservoir computing trends," *KI-Künstliche Intelligenz*, vol. 26, no. 4, pp. 365–371, 2012.

[12] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[13] S. Kauffman, S. A. Kauffman *et al.*, *At home in the universe: The search for laws of self-organization and complexity*. Oxford University Press, USA, 1995.

[14] D. Snyder, A. Goudarzi, and C. Teuscher, "Finding optimal random boolean networks for reservoir computing," in *ALIFE 2012: The Thirteenth International Conference on the Synthesis and Simulation of Living Systems*. MIT Press, 2012, pp. 259–266.

[15] S. Apostel, N. D. Haynes, E. Schöll, O. D'Huys, and D. J. Gauthier, "Reservoir computing using autonomous boolean networks realized on field-programmable gate arrays," in *Reservoir Computing*. Springer, 2021, pp. 239–271.

[16] Q. Xuan, Y. Li, and T.-J. Wu, "Growth model for complex networks with hierarchical and modular structures," *Physical Review E*, vol. 73, no. 3, p. 036105, 2006.

[17] S. K. Cherupally, ""hierarchical random boolean network reservoirs," Master's thesis, 2018.

[18] T. Kite, "Understanding PDM digital audio," *Audio Precision*, 2012.

[19] N. C. Sevuktekin, L. R. Varshney, P. K. Hanumolu, and A. C. Singer, "Signal processing foundations for time-based signal representations: Neurobiological parallels to engineered systems designed for energy efficiency or hardware simplicity," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 38–50, 2019.

[20] M. Bensimon, S. Greenberg, and M. Haiut, "Using a low-power spiking continuous time neuron (sctn) for sound signal processing," *Sensors*, vol. 21, no. 4, p. 1065, 2021.

[21] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," *Neural networks*, vol. 20, no. 3, pp. 335–352, 2007.

[22] M. Barthelemy, "Betweenness centrality in large complex networks," *The European physical journal B*, vol. 38, no. 2, pp. 163–168, 2004.

[23] S. H. Strogatz, *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

SMC-22 Paper Session 3

Kinesthesis: An Interactive Voice Software for Intertextual Improvisation in Theatre Performance

Tilemachos Moussas
Laboratory of Music Acoustics and Technology (LabMAT), National and Kapodistrian University of Athens
tmousas@minedu.gov.gr

Natalia Kotsani
School of Electrical and Computer Engineering, National Technical University of Athens
nkotsani@corelab.ntua.gr

Anasatasia Georgaki
Laboratory of Music Acoustics and Technology (LabMAT), National and Kapodistrian University of Athens
georgaki@music.uoa.gr

ABSTRACT

We present a novel human-centered gestural system for vocal improvisation, "Kinesthesis", to be used in new opera and musical theatre. We demonstrate the relationship between music composition, gesture and programming. An advantage of our system is the multimodal interaction technique through the invisible interface, which examines the use of electroacoustic techniques in human voice, in a theatre context. It explores the use of live or recorded, digitally processed voice as a sound source for the development of music cues for playback through a multi-channel speaker system. We also describe the application of the proposed system, in a real-time season theatre performance, where "Kinesthesis" was controlled through hand gestures from the actors, reinforcing the interactivity and highlighting the use of electroacoustic techniques in music theatre.

1. INTRODUCTION

Kinesthesia is a term that refers to bodily movement and perception. It is more specifically defined as, the sensation of bodily position, presence, or movement resulting chiefly from stimulation of sensory nerve endings in muscles, tendons, and joints [1]. Contemporary theatre is a deeply intertextual art form that weaves various performative elements within a narrative framework that is typically formed from existing cultural texts [2]. Intertextuality can be aesthetic, poetic and aleatoric, as it constitutes performance practices in creative performance. Performers display historical knowledge, situational competency, and sonify relations between them and their audience by bringing aspects of musical textuality into proximity through aural gestures. [3]

1.1 The Vocal Image in Post Dramatic Theatre

Nowadays one of the directions of music technology has been to increase the range of sound possibilities derived from the human voice and the level of efficient and flexible control over the voice. Specifically, the development of computer audio processing has made real-time sound processing possible and the communication between the performer and the device responsible for the sound manipulation.

For us humans, voice is a primary means of communication and expression, while all animals use voice to express a wide range of emotions and inner feelings. Humans rely on both linguistic and paralinguistic utterance as forms of communication and expression [4]. Paralanguage refers to the non-phonemic properties of speech, such as speaking tempo, tone, volume, rhythm and vocal pitch [4]. The kinesthetic sense, although not one of the commonly acknowledged five senses, is the subtle regulator of the complicated body patterns. The interdependent relationship between the breathing and phonatory mechanisms is fostered by kinesthetic awareness. [1]

An important role in modern approaches to musical drama is played by the "vocal image" which is understood as the creation or presence of a vocal sound that is intended to form an acoustic image either connected to the language or independent of the language [5]. The actor's resistance to a playful exploration of sound, in "non-language", has its roots in the levels of logic that are inherent in language. This resistance undoubtedly inhibits a dimension of vocal creativity as well as the possible vocal access to the articulation of the various texts of the post-dramatic theater. The intertextuality of the voice refers to a dimension of sound play that is already evident in theater and experimental theater practices. Sound and consequently voice, are aesthetic features, capable of making tangible and relevant, an invisible and complex force.

Famous contemporary composers created seminal pieces of music using electronic apparatus. Improvisation has always been present in musical performances, as a natural extension of interpretation [6]. In musical improvisation, data are structured in real-time as performers invent and control parameters, composing the music "in vivo". Improvisers follow the thread of their musically trained intuition into new and unknown musical contexts that the improviser might navigate in unexpected and novel ways.

1.2 The body as an instrument

According to Giorgio Agamben, "The gesture is communication of a communicability. It has precisely nothing to say because what it shows is the being-in-language of human

Copyright: © 2022 Author 1 et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

being as pure mediality" [7,8]. In other words: not only do gestures allow the body to talk without words, they also incorporate language; they embody language, and thus they are language and have a textual structure built into them. They communicate language, and they communicate by moving the body in space. The embodiment of language, and thus communicability would be nothing other than an incorporation of language into the human body [7].

In Greek ancient tragedy a highly formalised, anti-realistic acting technique utilises a pre-ordained vocabulary of gestures to indicate levels of emotional intensity. This language is based upon signs and not dependent upon words. A major characteristic of the Greek Drama is the exploitation of the full range of sounds, cries, and gestures, the use of facial masks, all these elements require the synthesis of artificial composition and creativity.

Through our research we approach the rich and complex ways that the voice acts inside the body but also outside the body of the speaker, when it is augmented in space during the performance. We study firstly the phonetic expression from the song's structured prosodic speech and secondly the deconstruction of the speech through the voice for the expression of the instincts according to the paralinguistic code. Psychologists and linguists have demonstrated that a person's voice pitch affects how others perceive her or him. [9] In our case A pitch shifter was used to reveal the performer's state of mind: an unusually high pitch may reflect agitation or an unchanging pitch may become boring or monotonous, decreasing the listener's span of attention. The pitch also could help the audience understand the performer's social status. For example, a person in authority position uses a higher pitch than the one used by a subordinate. The other parameter we used is the volume since the performer adjusts the volume of his voice depending on the size of the audience. The larger the audience, the louder the voice should be. Volume variation makes the speech effective. Sometimes changing from loud to soft and from soft to loud will have the desired effect. Mixed signals occur when the tone, pitch and gestures of the performer do not match with the words he is speaking. This combination, in addition to the use of the reverse delay, deconstructs the text and creates a paralanguistic code which is used to assist in the expression and the reflection of the performer's attitude. The result is a non-verbal material by nature which depends on voice, intonation, pitch, pause, volume, stress, gestures, and signals. Using our software the performer's voice can convey enthusiasm, confidence, anxiety and also the performer's mental state and temperament.

Kinesthesis can also extend the sonic palette via the exploration of non-human vocalisation. Spoken voice, shouts and singing voice produced by a vocalist are radically augmented in real-time by this custom interactive tool, giving the ability to the actor to create vocal soundscapes. Our aim is to expand and enrich the tools of both the drama actor, and the composer.

Plato called mimic arts those where the role of the artist is similar to a role of an instrument [10]. We focus in this "instrument" and how to increase its capabilities. The skillful use of the body as a musical instrument is the source of fine reciting an singing and only through heightened kinesthetic awareness can vocal skills be achieved and refined [11,12].

1.3 The Kinesthisis System

The motivation for developing Kinesthesis system, was the contribution in creating a new tool which converts the audience's speaking voice to singing voice, used in the participatory music theatre (new opera) "Oedipus"(2019), [13, 14]. Our system was used in a novel Theatre Documentary entitled "Kolokotronis's Memoirs". In this performance, the actor creates soundscapes with his voice and movements, but also improvises according to the dynamics of the moment, intervening and playing with space-time, thus sensually influencing the drama of the show and the listener himself, thus giving in each performance an unprecedented uniqueness. The concept of improvisation in the theatrical documentary suggests a non-essentialist approach to the concept of reality that is not restricted to mere representation. Therefore, the sound reveals the actions conversed and reshapes the linear narrative. There is a dramaturgy of sound, that rewrites another kind of text with the temporal and spatial dimension of acoustic events. [15] Also the voice is not merely as a means of speech, but an emotional, vibrant, gestural expression that transcends speech - and sound does not simply support music but is an autonomous stage building material.

The goal of creating Kineshesis was to implement a system that could give the performer the ability to: (i) control and process her/his voice wirelessly , (ii) offer continuous automation in adjusting the parameters using gestures in space, (iii) handle all the integrated effects simultaneously iv) provide multi-sensory stimuli to the audience through synesthesia

2. RELATED WORK

Twentieth-century performers pushed the limits of the voice's expressivity through varied and often astounding extended vocal techniques. Sound poets challenged the equivalency of language and semantic meaning by using phonemes as abstract sound. Composers combined the expressive and the communicative elements with the emerging electronic technology to further explore the limits to this potential of the voice. [16].

Homer Dudley introduced the "channel vocoder" (voice coder) in 1939, which operates on the principle of deriving voice codes to re-create the speech which it analysed. In the analysis stage, the fundamental frequency is determined and spectral information is provided by ten filters. The synthesizer discriminates voiced-unvoiced speech by use of the energy of the fundamental frequency and generates a buzz for voiced sounds and random noise for the hiss. This signal gets filtered corresponding to the analysis. Flanagan and Golden have been extending this model by taking the short-time magnitude and phase spectra of the signal into account. Since then the now called "phase vocoder" has been developed in various ways [17].

Research related in designing interactive voice softwares

for performance, started in 1949 with Pierre Schaeffer [18] and his "Symphonie pour un homme seul" which comprises eleven movements using recordings of vocal and other bodily sounds produced by a man [19]. Because of the voice's role as "primary expressive instrument", composers and performers alike have been drawn to an exploration of its expressive possibilities on multiple levels. [20]. In Alvin Lucier's work [21] "I am Sitting in a Room", he records himself speaking in a room recursively until the voice finally becomes incomprehensible and what the listener hears are the remnants of the voice as excitation of space [21]. The use of speech in the work, with stuttering and swearing, provided Lucier with the natural anomaly contained in the rhythms of speech [22]. Laurie Anderson is famous for the use of the vocoder in her work "O Superman" [22]. She generates an androgynous, robot-like vocal quality by channelling her voice through a vocoder to alternate between male and female speaking and singing voices. In Anderson's work "Stereo Songs for Steven Weed" [22], she creates a dialogue by turning her head between two microphones, controlling the placement of the voice in the stereo pair via her choice of microphone.

Pamela Z's has a growing body of installation works using multi-channel sound and video, triggering the sampled sounds with the Body Synth MIDI controller [23]. Hildegard Westerkamp often uses her works to reflect on her own experiences and beliefs. Both of these aspects are present in MotherVoiceTalk [24] where she uses the technological capabilities of the studio to create a musical space where negotiations are made between private and public, the self and the other, and time and space, resulting in a work whose blurred boundaries, allowing the listener an opportunity to reflect on his or her own stories before they are only like a memory. Simon Emmerson was also interested in "extending the timbral world of the live performer's voices and their acoustic instruments" [19]. In 1990 he created a work called Sentences for soprano and live electronics – the voice was processed live and distributed to the loudspeakers around the performance space. "If we consider vocal sounds, we refer here to the non-verbal, pre-linguistic possibly proto-linguistic exclamations: grunts, groans, moans, laughs, sighs, screams, and so on. These may express sadness, anger, awe, fear, satisfaction, humour and many other signals for sharing and communicating something about me or about the world" [25].

Donna Hewitt, is the inventor of the eMic [22], a sensor enhanced microphone stand for electronic music performance. By moving her body and the e-mic she distorts the sound waves, creating layers of music. Charmaine Lee's music is predominantly improvised, favoring a uniquely personal approach to vocal expression concerned with spontaneity, playfulness, and risk-taking. Beyond extended vocal technique, Charmaine uses amplification, feedback, and microphones to augment and distort the voice [26].

J. Cecilia Wu introduced the Expressionist, a system designed for electroacoustic vocal performers, who want to explore the possibilities of using body movement to enrich musical expression [27]. Expressionist uses a two-handed magnetic motion sensing controller and an interface that animates two- and three- dimensional interactive and dynamic environments that respond to hand motions captured by the 3D tracker. Laetitia Sonami, a sound artist, performer and composer, believes that the "novel gestural instruments have the potential to transform their creators" [28]. She developed the lady's globe [28, 29] and performed with her creation from 1991 to 2016, before moving to an object-based interaction, the Spring Spyre, constructed using a steel ring and magnets.

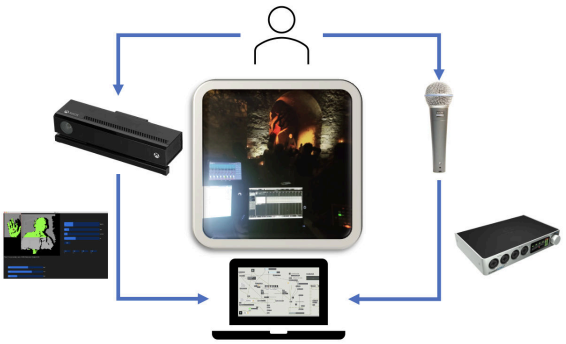


Figure 1. Kinesthesis' Architecture

The following works are using the kinect sensor in theatre performances. Schofield et al. created an augmented theatrical performance, called, Trigger Shift, following an 11-month long participatory design process with young participants [30]. Takatsu et al. implemented a system for supporting theatrical performances, that responds to speech and cues the next actor to speak [31] and Lycouris and Timmons introduced the 'Haptic Experiments' project which explores how blind dance audience members can experience the dynamic qualities of live dance performances through their sense of touch, using their hands [32].

3. ARCHITECTURE & IMPLEMENTATION

3.1 Architecture

Kinesthesis augmented voice tool, is a gesture control tool consisting of, as shown in Figure 1, (i) a wireless microphone to capture the voice, (ii) an external sound card, (iii) a computer, and (ix) the Kinect Sensor v1 [1]. It collects gestural data in real-time and controls an audio processing patch, which converts human voice to an augmented musical instrument. The microphone-captured audio is used as a source material while an arm gesture provides a computational input that is deciphered and reconfigured to trigger the sounds and controlling the effects' automation.

3.2 Sensor

The Microsoft Kinect sensor is a peripheral device that functions much like a webcam. However, in addition to providing an RGB image, it also provides a depth map. It is a Depth camera - Depth sensor. The library "open kinect for processing" was used in order to enable the Kinect

[1] In this implementation the Kinect 1414 was used, which is the original Kinect and works with the library documented on Processing 3.0 beta series.

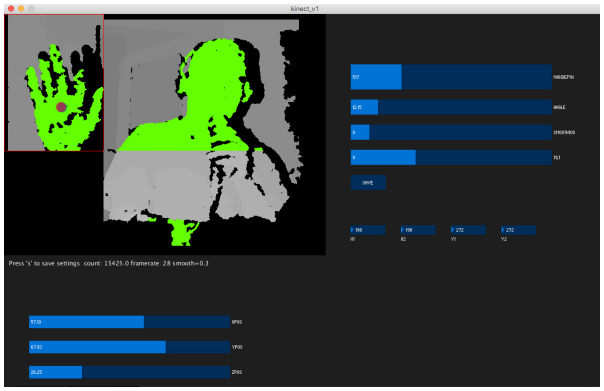


Figure 2. Max/MSP and Sensor's Integration

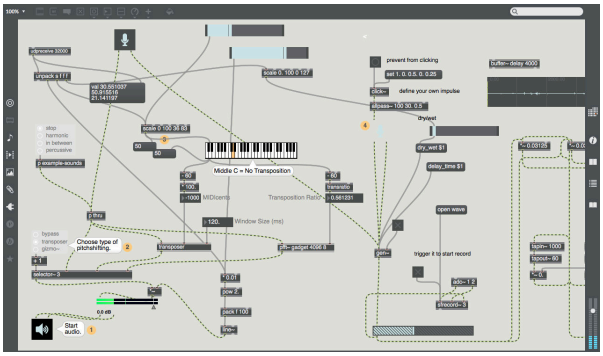


Figure 3. Kinesthesis Max/MSP Patch

to be used with Mac OS. It uses the open source driver "Libfreenect" to connect the computer with the Kinect. Libfreenect allows us to retrieve RGB images, depth maps, images from the infrared camera, accelerometer readings and also controls the Kinect's tilt motor and to use multiple Kinects simultaneously (depth sensors may interfere with each other) [33]. What we built for processing allows us to get the raw data from the Kinect. It has three eyeballs: a) an infra-red projector which sends out infrared light into the room, b) a sensor - camera that reads the infra-red light. The kind of light which passes out is a whole lot of infra-red dots into the room. (there is a specific pattern of dots that Kinect can detect. c) an RGB camera (each pixel has a red /green and blue value). So if this surface is curved the dots appear distorted, then it distinguishes the things that are closer from the things that are further away, measuring the distance from the sensor. The Kinect measures in mm and the raw depth values range between 0 and 2048. The range of depth that the kinect(v1) can see is 0.7–6 meters or 2.3–20 feet.

3.3 Max/MSP and Sensor's Integration

A free graphical library and integrated development environment, "Processing 3.0" (in Java programming language) was used. We added the library "Open Kinect for Processing", to iterate over the raw depth data array and the PVector which is a three dimensional vector, specifically a Euclidean (also known as geometric) vector and has both magnitude and direction. This datatype stores the components of the vector (x, y, z coordinates). The OSCP5 library was used to send OSC (Open Sound Control) messages to the Max/MSP patch. The Max/MSP patch receives the OSC messages transmitted over a network using the User Datagram Protocol (UDP). Finally, the list of data disagregated into three individual messages, each one for one of the three parameters we want to process, coordinate x for the volume, coordinate y for the pitch (4 octaves) and coordinate z for the percentage of the effect, in this specific case, reverse delay. We use our hand gesture on an xyz axis to control the above sound parameters in real time. The implementation can be found here [2].

4. INTERFACE

Kinesthesis is an accessible interface for the real-time structuring of data and gives to the performer the opportunity to escape the limitations of desktop interaction with musical controllers. The performer does not need a video monitor provided, that we have already calibrated the sensor.

In the Kinesthesis simple interface shown in Figure 3, the user can choose the type of pitch shifting (bypass, transposer or gizmo) and optionally the type of midi instrument he/she wants to follow his/her voice (harmonic, in between, percussive), or the "stop" selection to mute the midi instrument. The user can also select one of the five different types of reverse delay (presets). Reverse delay works mainly with these particular objects: a "buffer" (which feeds the audio data, through a short recording of the voice) and a "gen" that has the ability to accept different "in" objects and mixes them with two different signal streams. We use "poke" and "peek" objects to accurately reproduce within a few milliseconds the recorded phrase which automatically is being played backwards because of the use of the "counter" object which accumulates and exports a stored amount of data. The "counter" is being activated at a sampling rate through "dim" (the third input) and is updated through the "param" object. The feedback of the whole code is being achieved, by using the "history" object.

The Kinect sensor, must be positioned in front of the performer (2 meters distance) and a calibration of the Kinect via the Processing sketch is required (MaxDepth, Angle, Smooth100, Tilt, taking account of the performer's height, size, position, and distance from the sensor. The user can select whole body or to focus on the hand gestures (Figure 2).

5. EVALUATION

We used the following ways of augmenting the voice [3] : a) Singing, b) Singing with speech in specific tonal heights (Sprechstimme). c) Speech in vague tonal heights in high, medium and low range with specific duration. (Used to describe specific effects, such as screams and exclamations.) d) Verbal way without tonal heights with specific duration. (It is essentially a recitation in a strict rhythmic context.)

[2] drive.google.com/drive/folders/1lvHRnCYSpcSYxOkYKU2CWAcSfO35tWgf?ths://www.youtube.com/watch?v=kD98x8NOZn4

e) Combinations of the above ways, so that a sentence begins vocally and ends verbally and vice versa. (It is also possible to switch from one type of recitation to another.)

The response of the system was immediate and efficient.The control of digital effects did not show any delays, and there were no problems during the implementation. The test was performed on a MAC OS X operating system, using a Shure Beta 58A microphone, an iConnectAUDIO4+ sound card and the Kinect V1.

6. THEATRE PERFORMANCE

Kinesthesis augmented vocal tool was applied in the interactive theatrical documentary "Kolokotronis Memoirs" in 2021 for 21 performances from 27th of July till the 16th of August and took place in the History Museum of National and Kapodistrian University of Athens (Figure 4). "Kolokotronis Memoirs" is a research project, of the Laboratory of Music Acoustics and Technology (LabMAT) of the National and Kapodistrian University of Athens. The project was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.).

The stage narrative places the Greek general, Kolokotronis, a symbol of the Greek Revolution, in a peculiar, imaginary, historical museum, as a living and present "exhibit". Focusing on the theatrical form of Kolokotronis's speech, the play narrates his life and deeds, while at the same time the narrative is framed by soundscapes, live music, video art and live filming using digital technology, in a continuous dialogue between the historical past, memory and uptake into the present. Three actors and a musician (clarinet player), performed live. The main character, was creating soundscapes with his voice and his imposing hand gestures, using Kinesthesis, and creating a communication channel between the media in a heterotopic pursuit of history.

The protagonist, was enthusiastic using Kinesthesis system. It took him only several rehearsals to feel comfortable with it. His voice was captured by a lavalier wireless microphone. While he was a little bit insecure using the reverse delay effect, he was enjoying improvising with the pitch shifting and the change of the volume dynamics of his voice. However he was composing aleatoric soundscapes which were coming from an unfamiliar choreography created by his hand gestures. The performance was full house everyday and had great reviews, Kinesthesis transformed the actor to an interpreter of the acoustic memories of the Greek revolution, in a highly orchestrated way.

7. CONCLUSION

Performers need to be able to use the technology and to operate new tools without having any specialized technological knowledge. The tool we have developed is quite simple and unsophisticated in its use, when compared with other (previous mentioned) systems in interactive art, however, Kinesthesis is a versatile, robust, easy to assemble, low-cost tool that offers a new aesthetic approach for interactive sound dramaturgy in New Opera and immersive

Figure 4. Kinesthesis in Kolokotronis Performance

music theatre. Real time voice processing expands the expressive theater language, giving the director more tools to show the inner conflicts of the character, the opposing moments of action / thought, the memories, the worries, the desires.

"As the tongue speaketh to the ear, so the gesture speaketh to the eye" as Sir Francis Bacon said. [34]. Specifically, in gesture, sound can be used to represent a performer's expressive movements in the same medium as the performed music, making relevant visual cues accessible through simultaneous auditory display. "As the tongue speaketh to the ear, so the gesture speaketh to the eye" Using Kinesthesis the performer "manipulates" the voice as material in real time, so that he/she is both a performer and a director and a composer. "An augmented or hyper instrument is an enhanced traditional instrument with various sensors to enable features of the gestural activity of performers to control augmentations of the existing instrumental sound" [35].

8. FUTURE WORK

The enrichement of Kinesthesis with more effects, harmonizers and predefined gestures, is under development. Also, a future extension of this work will be the incorporation of the Omax created by the IRCAM [36], so as to enable the actor to utilize live or pre-recorded material and to interact and co-improvise with the computer. Thus Kinesthesis will turn to be a media actor, itself.

Acknowledgments

The research work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant" (Project Number: 150).

We would like to thank Giorgos Dedousis for his valuable contribution in the realization of the "Kolokotronis Memoirs" performance. We would also like to thank Yiannis Kranidiotis for his assistance in the processing sketch, Coti K for his guidance and Penelope Bekiari for her ideas.

9. REFERENCES

[1] K. T. Bauer, "The role of kinesthesia in a pedagogy for singing," *Journal of Singing-The Official Journal of the National Association of Teachers of Singing*, vol. 73, no. 2, pp. 129–136, 2016.

[2] A. C. Rush, K. Jones, R. Dean *et al.*, "Recycled culture: the significance of intertextuality in twenty-first century musical theatre," Ph.D. dissertation, University of Lincoln, 2017.

[3] J. Strachan, "Reading ascension: Intertextuality, improvisation, and meaning in performance," *Critical Studies in Improvisation/Études critiques en improvisation*, vol. 9, no. 2, 2013.

[4] B. Truax, "Soundscape composition as global music: Electroacoustic music as soundscape," *Organised Sound*, vol. 13, no. 2, pp. 103–109, 2008.

[5] E. A. L. Mendes, "A study with hyperinstruments in free musical improvisation," *NICS Reports*, no. 14, 2016.

[6] A. L. Berkowitz and D. Ansari, "Expertise-related deactivation of the right temporoparietal junction during musical improvisation," *Neuroimage*, vol. 49, no. 1, pp. 712–719, 2010.

[7] M. Hallensleben, "Introduction: Performative body spaces," in *Performative Body Spaces*. Brill, 2010, pp. 9–27.

[8] G. Agamben, "Notes on gesture," *Means without end: Notes on politics*, vol. 20, 2000.

[9]

[10] I. Murdoch, "From the fire and the sun: Why plato banished the artists," in *Plato on Art and Beauty*. Springer, 2012, pp. 3–33.

[11] M. Lotze, "Kinesthetic imagery of musical performance," *Frontiers in human neuroscience*, vol. 7, p. 280, 2013.

[12] S. L. Foster, "Movement's contagion: The kinesthetic impact of performance," *The Cambridge companion to performance studies*, vol. 46, 2008.

[13] "Oedipus: Sex with mum was blinding," https://arts.stanford.edu/event/83280/, 2019.

[14] "Oedipus: Sex with mum was blinding," https://www.bam.org/oedipus, year = 2019.

[15] D. Z. Saltz, "Live media: Interactive technology and theatre," *Theatre Topics*, vol. 11, no. 2, pp. 107–130, 2001.

[16] L. Barkhuus and C. Rossitto, "Acting with technology: rehearsing for mixed-media live performances," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 864–875.

[17] F. Hammer, "Time-scale modification using the phase vocoder," *Diploma Thesis*, 2001.

[18] T. D. Taylor, *Strange sounds: Music, technology and culture*. Routledge, 2014.

[19] S. Emmerson, "Electroacoustic music before language." EMS Network, 2018.

[20] N. Collins and J. d'Escriván, *The Cambridge companion to electronic music*. Cambridge University Press, 2017.

[21] A. Lucier, A. Lucier, A. Lucier, and A. Lucier, *I am sitting in a room*. Lovely Music, 1970, vol. 1013.

[22] D. Hewitt, "Compositions for voice and technology," Ph.D. dissertation, University of Western Sydney (Australia), 2006.

[23] G. E. Lewis, "The virtual discourses of pamela z," *Journal of the Society for American Music*, vol. 1, no. 1, pp. 57–77, 2007.

[24] A. S. J. McCARTNEY, "Sounding places: Situated conversations through the soundscape compositions of hildegard westerkamp," Ph.D. dissertation, Verlag nicht ermittelbar, 1999.

[25] J. Chadabe, "The electronic music foundation," *Organised Sound*, vol. 2, no. 1, pp. 25–27, 1997.

[26] P. Rodrigues and M. Vairinhos, "Integrating interactive multimedia in theatrical music: the case of bach2cage," in *Artech 2005-2º Workshop Luso-Galaico de Artes Digitais Vila Nova de Cerveira, Portugal 27 de Agosto de 2005 Livro de Actas*. International Association for Computer Arts, 2005, p. 111.

[27] J. C. Wu, "The expressionist: a gestural mapping instrument for voice and multimedia enrichment," *The International Journal of New Media, Technology and the Arts*, vol. 10, no. 1, pp. 11–19, 2015.

[28] M. Mainsbridge, *Body as Instrument: Performing with Gestural Systems in Live Electronic Music*. Bloomsbury Publishing USA, 2022.

[29] L. Sonami, "On my work," *Contemporary Music Review*, vol. 25, no. 5-6, pp. 613–614, 2006.

[30] T. Schofield, J. Vines, T. Higham, E. Carter, M. Atken, and A. Golding, "Trigger shift: participatory design of an augmented theatrical performance with young people," in *Proceedings of the 9th ACM Conference on Creativity & Cognition*, 2013, pp. 203–212.

[31] R. Takatsu, N. Katayama, T. Inoue, H. Shigeno, and K.-i. Okada, "A wearable action cueing system for theatrical performance practice," in *International Conference on Collaboration Technologies*. Springer, 2016, pp. 130–145.

[32] S. Lycouris and W. Timmons, "(choreo)-haptic device: Moot digital transformations," in *MOOT: Digital Transformations*, 2012.

[33] S. Kean, J. C. Hall, and P. Perry, "Kinect for creatives," in *Meet the Kinect*. Springer, 2011, pp. 85–99.

[34] R. M. Krauss, "Why do we gesture when we speak?" *Current directions in psychological science*, vol. 7, no. 2, pp. 54–54, 1998.

[35] J. Bowers and P. Archer, "Not hyper, not meta, not cyber but infra-instruments," in *Proceedings of the 2005 conference on New interfaces for musical expression*. Citeseer, 2005, pp. 5–10.

[36] G. Assayag, G. Bloch, and M. Chemillier, "Omax-ofon," in *Sound and Music Computing (SMC) 2006*, 2006, pp. 1–1.

FPGA-accelerated Real-Time Audio in Pure Data

Clemens Wegener
Interface Design Group
Bauhaus-Universität Weimar
clemens.wegener@uni-weimar.de

Sebastian Stang
The Center for Haptic Audio Interaction Research (CHAIR)
sebastian@chair.audio

Max Neupert
The Center for Haptic Audio Interaction Research (CHAIR)
max@chair.audio

ABSTRACT

With the advent of fast ARM processors more audio products are running embedded Linux systems and using high level languages to implement real-time signal processing. Still, embedded Linux systems can't compete with the processing power of desktop computers to implement complex signal processing as needed for physical modeling algorithms. This paper describes how to interface custom digital logic circuits in an Field Programmable Gate Array (FPGA) with the Linux operating system to speed up processing. The hardware used is a Terasic DE10-Nano development kit equipped with an Intel [1] Cyclone V SoC. [2] We are running Pure Data (Pd) on Linux and communicating with a mass-interaction network for physical modeling sound synthesis on the FPGA. The code referenced in this publication is available online. [3]

1. INTRODUCTION

1.1 DSP on embedded Platforms—a brief Overview

The landscape of available hardware platforms capable of running digital audio applications is diverse and ranges from microcontrollers to full Linux systems running on multi core CPUs. To pick the right platform for a project is an optimization challenge to get the most computing power for as little resources as possible. But not only that: In most use-cases additional specifications are important. A fan-less, passively cooled system is usually required and—if used as an instrument—it's mandatory that latency and jitter of the input to output pipeline stays in tolerable [1] limits. For light projects, platforms without the overhead of an operating system have the advantage to be affordable, compact, and robust. The Axoloti [2], Electrosmith Daisy, and Teensy [3] boards are examples of this category. When it comes to compact Linux boards, the choices are almost endless. Leaving industrial single board computers and SOMs [4] aside, community friendly options are mostly ARM architectures which include the Pine64 or Odroid boards. The Raspberry Pi is in this group and is arguably the most popular choice. To optimize Linux systems for real-time audio and low latencies there are two main strategies:

1. Using a PREEMPT_RT [4] kernel or a low-latency kernel with *rtirq*, assigning the audio processes higher priority [5] in the scheduling policy, reserving a processor core for the audio processes; all the usual Linux audio optimizations. Patchbox OS by Blokas [6] and Satellite CCRMA [5] are good examples for embedded Linux systems preconfigured that way.

2. The more sophisticated solution achieving supreme results at the cost of less portability involves using a Xenomai kernel. One design option is to remove the Audio IO from the kernels tasks completely to give it higher priority in the user space above the kernel. This is how the Bela [6] features outstandingly low latencies sitting on top of the Beaglebone Black. Elk Audio OS uses the Xenomai 4 dual kernel architecture [7] with a general-purpose kernel and a dedicated real-time kernel. Elk Audio [7] is designed to run on a Raspberry Pi system. Bela and Elk Audio both depend on custom hardware with audio codecs and additional features which attaches to their respective computing platform.

A comparison between the Xenomai based Elk Audio and a PREEMPT_RT system running on the same hardware (Raspberry Pi) can be found in Vignati et al. [8]. A third option would be to use a Linux OS capable processor, but running the process "bare metal", treating the processor like a microcontroller without the overhead of an operating system. This allows for an impressive embedded audio performance. An example of this technique on the Raspberry Pi is hinted to in Michon et al. [3]. The drawback is that it makes development more complex if there is need for OS features. Additional interfaces, graphics display, and everything that the OS would be already providing would need to be implemented once again.

[1] Intel acquired and absorbed Altera in 2015 re-branding the Altera products to the Intel trademark.

[2] *System-on-Chip:* an integration of core components of a computer in one chip (integrated circuit).

[3] https://github.com/chairaudio/SMC22-FPGA-accelerated-PD

[4] *System on Module*: a compact circuit board, typical for embedded systems. Usually populated with a processor or microcontroller and memory. It is connected to a host board where peripherals can be attached.

[5] By setting the "nice value".

[6] https://blokas.io/patchbox-os

[7] https://evlproject.org

Copyright: © 2022 Clemens Wegener et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

A third category—besides fast micro controllers and low-latency OSs—leverages the computing power of FPGAs. Pfeifle and Bader [9] show, that FPGAs are particularly strong in computing physical models in the form of finite difference schemes, because these algorithms lend themselves to parallelization and their need for local storage variables is modest. For the usage in musical instruments FPGAs have two major drawbacks: designs need long development times and usually solve domain specific problems. E.g.: a design that models a vibrating membrane is inefficient in computing a string. Verstraelen et al. [10] summarize this problem as follows:

> Our conclusion is that FPGA technology has the ability to implement computational intensive real-time physical models of musical instruments, but the problem is to make this technology sufficiently flexible.

The usage of High Level Synthesis (HLS) languages as in Vannoy et al. [11] addresses the problem of long development times by using abstract behavioral models that automatically translate to complex digital logic designs and hiding away implementation details to the developer. Risset et al. [12] refine HLS by automatically translating FAUST programs—a data flow language specifically written for DSP programming—to FPGA digital logic. This is a very promising approach to speed up the development process in the context of musical applications and it would allow a broad library of open source code to use the massively parallel computing power on FPGAs with very low latencies.

But it does not ease one of the drawbacks of fixed digital logic architectures: It would be desirable to program FPGAs as flexible as software running on a CPU: algorithms or effect chains running on digital logic should be dynamically changed and be re-configurable by the user. I.e., it is not possible to load libraries of compiled code on an FPGA—while the system is running—to change the behavior of the circuit in a way that the designer has not foreseen. [8]

Verstraelen et al. [13], [10] approach this problem by designing a DSP processor that is tailored to compute a wide variety of physical models efficiently, without the need to change the architecture of the digital logic. Their solution is the design of a specialized DSP processor that uses the advantages of FPGA's distributed and massively parallel computing capabilities. This processor can be instantiated on an FPGA or be commercially produced as an ASIC. [9] On top of this DSP processor runs software written in a custom declarative language (conceptually a HLS language again) but comes with the benefits of faster compilation and re-programmability, similar to software on general purpose CPUs or DSPs. The application running on this DSP could theoretically be designed to be re-programmed in real-time, changing its data flow while audio is streaming through it, by inserting parts of pre-compiled code.

1.2 Motivation for this Research

Our motivation to investigate beyond the obvious choices of embedded platforms mentioned earlier was to create a unique platform for our instrument [14] enabling real-time acoustic excitation of complex physical models, possibly implemented in lumped mass-spring networks. This led to our wish for a powerful system which can compete with or even outperform desktop performance in key aspects. At the minimum it should offer *some* technological feature which is unique to the platform in comparison to the rest of the embedded system landscape. We briefly investigated parallel processing systems and therefore acquired an Nvidia Jetson Nano with the hope to implement physical models in shaders as discussed in Zappi et al. [15]. At the same time we also began to investigate the ARM-FPGA-SoC route and finally decided to focus on the latter due to the overall flexibility of the system, such as being able to integrate several audio input and output streams in digital logic design.

FPGAs are extremely powerful devices which can be configured by the user. This allows the FPGA to instantiate logic circuits like processors [10] which are not a mere emulation but truly identical to the original hardware down to the logic blocks. For this reason our particular development board is very popular in the retro-gaming scene. Gamers can experience arcane gaming platforms with the exact same performance and glitches like the original hardware. FPGAs are available with different amounts of logic cells and operators and can cost anywhere from 90 ct to 150.000 €. [11] Common uses for FPGAs are neural networks, simulations, database queries, LED-matrix controllers, prototyping of integrated circuits in hardware development, and cryptology. For DSP-applications a sufficient amount of available multiplier cells in the FPGA is essential. FPGAs can replace DSP chips and are also used in audio interfaces with hardware effects. Complex audio computations can be calculated massively parallel in FPGA logic with very little processing latency.

Since the Intel Cyclone V SoC combines FPGA and ARM processor in one chip, we can use the ARM CPU to run Pd. Pd is a graphical real-time programming environment. Using Pd in the product was a design decision to allow us to draft and deploy in the same language. This is liberating us from the complication of translating the entire algorithm from Pd to FPGA logic. Instead we can make better use of the FPGA focusing on specialized tasks which would be computationally expensive to implement on the CPU. Such computations can be a mass-spring-model, a large reverberation effect or similar DSP blocks which are then represented as an atom in the graph of a Pd patch. An additional benefit of using Pd is that the system can be programmed in real-time: that allows for fast prototyping and "tuning algorithms by ear", while the system is running.

[8] It can be argued that a part of the flexibility of software can be mimicked by a technique called *partial reconfiguration*, where parts of a logic design can be replaced. But—unlike in software—this method is limited by the hardware resources that the predefined reconfigurable area provides.

[9] *Application Specific Integrated Circuit*. ASICs are developed to solve domain specific problems and have medium to high volume productions.

[10] Processors implemented in FPGA hardware are called "soft cores".

[11] Based on a quick search on the website of a supplier. Many chips were not even available, probably due to the ongoing shortage.

We see "tinkerability" in a potential product as a great feature. In the musical instruments market it is not uncommon that the environment is open to the user so it can be tailored to specific needs or features contributed by the community. Products like the Organelle or the multi-effect devices from MOD-Devices are illustrating this paradigm. Exchanging parts of a DSP graph between the clocked FPGA and scheduler based Linux does however introduce complexity and comes with challenges for the timing of the processes. This paper is written to address these challenges and evaluate the proposed solutions.

Miller Puckette announced that he intends to port Pd to FreeRTOS so that it can run on the Espressif LyraT (ESP32) board. [12] Running Pd on FreeRTOS may further facilitate the FPGA integration and remove most of the obstacles caused by the different computing approaches. We look forward to test this when it becomes available.

The motivation for this research was to answer the following questions: "Is it possible to include an FPGA coprocessor in the signal processing chain while using a proven audio programming environment (Pd) on a Linux OS? Will the performance of such system be sufficient for physical modeling synthesis and the latency acceptable?"

2. HARDWARE AND SOFTWARE COMPONENTS

We chose the DE10-Nano as a platform because it's one of the entry products which nevertheless features a reasonably sized FPGA along with an ARM core for running Linux. The physical modeling algorithms we intend to run, require sufficient hardware multipliers and on-chip ram. The DE10-Nano features 112 fixed point 27 Bit multipliers, which can run at a clock rate of 350 MHz. The ARM is a dual core running at 800 MHz clock rate each.

The DE10-Nano board does not have an audio codec, therefore we had to install it separately. We decided for a Mikroe-506 board featuring a Wolfson WM8731 codec. Another component that is missing on the SoC is the I2S interface to stream audio data. We build upon work of Bjarne Steinsbø for Terasic's DE1-SoC who wrote both—the I2S hardware interface and the accompanying opencores-I2S device driver. [13] This gives us Verilog code to instantiate the necessary hardware I2S interface inside the FPGA fabric (see Fig. 1).

For the codec chip itself we configured the Altera Linux Kernel 5.4.54-LTS to include the Wolfson 8731 codec driver and added support for the ALSA [14] System on Chip (ASoC) simple card. [15] The ASoC simple card driver glues together the opencores-I2S device with the WM8731 codec and is the interface between the OS (ALSA) and these components (See Fig. 2).

Finally we installed Pd v.0.51.2 with ALSA support on the provided Ångström v2016.12 Linux console image. The custom digital logic for the sound card is completely opaque to user space software. We can simply use the standard ASoC simple card as an input and output device. This allows for a standard user experience in Pure Data (or any other audio software on the platform).

On the other hand, the DSP subsystem that is instantiated in FPGA logic needs a non standard approach for communication and audio data exchange. To address this issue, we wrote a Pd external that sends and receives audio blocks and communication data with the FPGA (see section 3.2).

3. DATA FLOW BETWEEN PURE DATA AND FPGA

On this SoC platform, FPGA and ARM share the same die—this is beneficial for fast data transfers at low latency. The Cyclone V SoC features an industry standard AXI bridge to allow high speed communication between both devices.

3.1 Memory Mapping

The AXI bridge is at a hardware address that is normally only visible to the kernel. To access this region from user space, we map the hardware address from `/dev/mem` to our local program memory using `mmap()`. This sends a request to the kernel to set up page tables so we can redirect to hardware addresses. For this we need read/write privileges to `/dev/mem`. On the FPGA side we can define memory interfaces and other peripherals to be mapped to sub-addresses of the AXI bridge. To read and write data to the bus, we can then just map the hardware (sub-) addresses from the AXI bridge to our local program memory and access it through a pointer.

In our case, the physical memory address of the Heavy Weight AXI bridge is given to be `0xC000_0000`. On the FPGA side we instantiated several on-chip memory interfaces as shown in Fig. 3. The AXI bridge sub-addresses are automatically offset to local addresses in the FPGA memory space, so we can directly read and write to them in a C program as if they where local arrays.

3.2 Pure Data External

Pd is a real-time data flow programming environment. It uses graphical objects ("atoms") which typically have sinks ("inlets") and sources ("outlets") that allow for interconnections to route data or signals in the graph. Pd ships with a number of core objects that solve standard problems. The user can extend these objects by "abstractions", which are itself written in Pd, similar to writing a reusable function in other programming languages. Objects written in C and compiled for Pd are called "externals". Libraries of abstractions and externals mostly maintained by third parties can be distributed via Deken; Pd's integrated package manager.

Enabling the FPGA to exist as a node of the DSP graph of Pd, we wrote an external which takes care of copying the data from the CPU to the FPGA and back.

To allow audio signals and control data to be sent from Pd to the FPGA on-chip memory we use the memory mapping technique described above. Blocks of audio samples that arrive at the externals input can be directly written to the

[12] PD-dev Mailinglist 2021-07-13 `https://lists.puredata.info/pipermail/pd-dev/2021-07/022773.html`

[13] The source code is available at `https://github.com/bsteinsbo/DE1-SoC-Sound`.

[14] *Advanced Linux Sound Architecture*

[15] Available at `https://github.com/altera-opensource/linux-socfpga`.

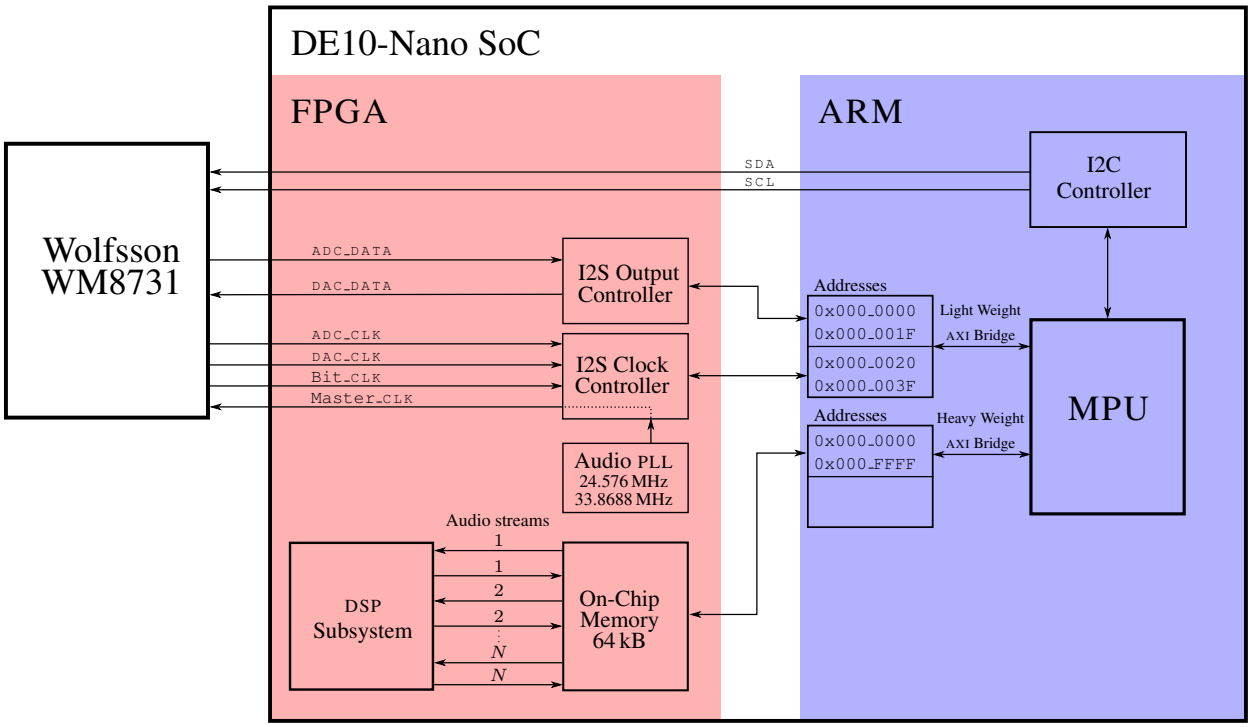


Figure 1. Hardware Components.

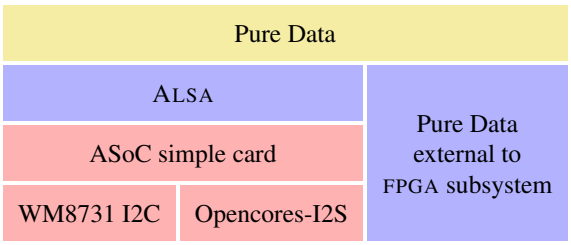


Figure 2. User Space, Kernel Space, Kernel and User Space

```
 ( PROT_READ | PROT_WRITE ),
 MAP_SHARED, //synchronize data
 x->fd, 0xC0004000); //hardware address
x->in_smp_buf_ptr=(int *)(x->in_smp_buf_map);
// mapping output sample buffer
x->out_smp_buf_map = mmap( NULL,
 OUT_SMP_BUF_SPAN,
 ( PROT_READ | PROT_WRITE ),
 MAP_SHARED,
 x->fd, 0xC0005000); //hardware address
x->out_smp_buf_ptr=(int *)(x->out_smp_buf_map);
```

Listing 1. Mapping on-chip memory to the local sample buffers.

memory mapped arrays and will then be sent over the AXI bridge to FPGA on-chip memory.

In the external's constructor we need to set up the memory map and retrieve a pointer to our hardware for later use as shown in listing 1. In the DSP routine of the external (the DSP callback) we can then use the pointer to read and write samples to this buffer as shown in listing 2. Note that we can detect dropouts, by reading and clearing a flag that the digital logic on the FPGA sets when it is finished processing the samples. We can also tell the FPGA how many samples it should process by writing this information to the on-chip memory. This way, we can change our block size in the Pd patch and the FPGA will immediately follow the change, shrinking or increasing the amounts of samples to process dynamically.

```
// opening /dev/mem for mapping to local mem.
x->fd = open("/dev/mem", (O_RDWR | O_SYNC));
// mapping input sample buffer
x->in_smp_buf_map = mmap(NULL,
 IN_SMP_BUF_SPAN, //size of map
```

```
...
// data struct for external
t_fpga_tilde *x = (t_fpga_tilde *)(w[1]);
// input samples to external
t_sample *audio_in=(t_sample *)(w[2]);
// output samples from external
t_sample *audio_out=(t_sample *)(w[3]);
// current block size
int n = (int)(w[4]);

if (x->out_smp_buf_ptr[0] != 0xabcd) {
   // fpga not finished
   pd_error(x,
     "FPGA buffer dropout detected");
 }
// copy data to fpga
memcpy((void*)&x->in_smp_buf_ptr[512],
  (const void*)audio_in,
  4*n);
// get old (processed) data back from FPGA
memcpy((void*)audio_in,
  (const void*)&x->out_smp_buf_ptr[512],
  4*n);

// tell FPGA number of samples to process
x->in_smp_buf_ptr[1] = n;
// pure data: buffer ready flag
x->in_smp_buf_ptr[0] = 0xabcd;
// clear FPGA ready flag
```

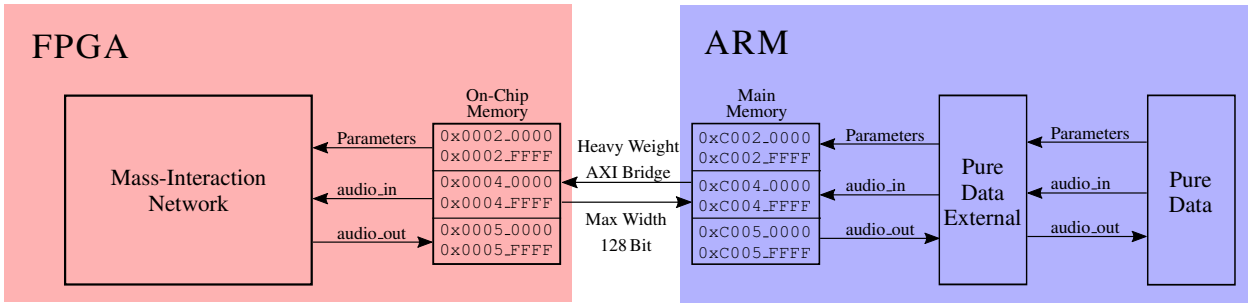


Figure 3. Data flow between Pd and FPGA.

```
x->out_smp_buf_ptr[0] = 0x0;
...
```

Listing 2. DSP callback function of the Pd external. Using flags, FPGA dropouts can be detected.

3.3 FPGA Digital Logic

On the FPGA side, digital logic was written to read out the on-chip memory modules and feed the data forward to other DSP processes.

Index	ARM Data	FPGA Data
0	ARM Ready	FPGA Ready
1	Block Size	Buffer Interval Count
2-511	Reserved	Reserved
512-1023	Samples	Samples

Table 1. Data structure of the buffer exchanged with the FPGA. Data size is 32 bit.

For measuring jitter and system latency, we implemented a state machine that waits for new data, and then simply copies the audio input to the audio output buffer and sets a flag when it is done. An additional counter to measure the time interval between two buffer arrivals was also implemented. Table 1 shows the structure of exchanged buffers. The ARM and FPGA buffers each reside in a dedicated on chip RAM as depicted in Fig. 3.

Our real-world DSP application is a mass-spring model, which unfortunately will have to be described in another paper since we hit the 8 pages limit here.

3.4 Buffer Transfer Speed

Fig. 3 shows how data flows between Pd and the FPGA portion of the SoC: Audio samples, parameter- and control data is sent through the *Heavy Weight AXI Bridge* which is the fastest hardware interface available on the ARM SoC. This bus is 128 Bit wide and runs at a maximum speed of 200 MHz. This leads to a theoretical maximum bandwidth [16] of 3.2 GB/s between FPGA and ARM. [16]. However, the achievable speed varies with the applied memory access techniques, the OS and the transferred data sizes.

[16] The bridge is 128 Bit wide, with a maximum clock speed of 200 MHz.

On Ångström embedded Linux a buffer of 64 32-bit integers transfers happen to be similar for `memcpy()` transfers and DMA transfers—they are between 60-90 MB/s fast [16]. For bigger buffer sizes, using DMAs is significantly faster: A 256 samples buffer has a transfer rate of 250 MB/s using DMA and 60 MB/s using `memcpy()`. Top speeds for DMA transfers are at 8192 samples with 600 MB/s throughput while `memcpy()` peaks out at 60 MB/s. Xilinx Zynq platforms have similar transfer speeds in our system configuration (see [17]).

Bruce R. Land reports read/write rates of 28 MB/s using single element access (no `memcpy()`) and 266 MBytes/s using DMA for a buffer size of 10.000 32-bit integers. [17]

4. CONFIGURING PURE DATA FOR REAL-TIME BUFFER TRANSACTIONS TO FPGA HARDWARE

A particular challenge in real-time systems is that information needs to be passed just in time to form a continuous data flow. We need to make sure that buffers are passed from Pd to the digital logic design at regular intervals. When intervals vary, the amount of time that can be spend for computational tasks in the DSP subsystem will vary too.

However, operating systems are usually optimized to solve concurrent tasks and prioritize those that the scheduler assumes to be critical. Running programs may signify the scheduler that they finished their task by *sleeping*. After either a predefined interval or an external wake up mechanism they will be called to work again, but must compete with other tasks for computational time. This leads to the situation that data is typically processed in irregular intervals, based on the arbitration of the scheduler.

Pd was written to run on non-real-time systems and is capable to deal with task scheduling by the OS—essentially allowing a balancing of computational load between its audio tasks and parallel tasks of the OS.

Pd handles this load balancing with its own internal scheduler: By default it computes the message and audio flow for blocks of 64 samples each [18, p. 63]. When the computation of one block of message and audio data flow is completed, it *sleeps*, which allows the OS's scheduler to delegate the free compute time to other processes. The

[17] `https://people.ece.cornell.edu/land/courses/ece5760/DE1_SOC/HPS_peripherials/FPGA_addr_index.html`

sleep interval in Pd's scheduler is calculated to be of "reasonable" size (Pd's global delay setting divided by four). In our case this was 0.75 ms. [18] That means, when the last block of audio was processed and no more data is in the pipeline, the scheduler sleeps [19] for 0.75 ms and frees up the processor. During that time a couple of new blocks (here, a block of 16 samples had a real-time duration of 0.33 ms) accumulate in the processing pipeline and will get processed in a fast burst when Pd awakes from sleep.

This burst processing behavior is undesirable in the presence of external processes that need to be synchronized to each sample block. When a burst occurs, the external process (in our case the digital logic on the FPGA) has very little time to compute the sample block before it must be returned to the main process. Fortunately Pd provides means to reduce sleeping time and thus get more predictable block process intervals. By adjusting the *sleepgrain* parameter—either through a startup flag or a Pure Data external—the timing of block calculations can be altered. In short: the smaller the *sleepgrain* size, the less jitter will occur on the block processing intervals but the more function calls and context shifts between the OS and Pd will occur that will slow down other processes running on the OS.

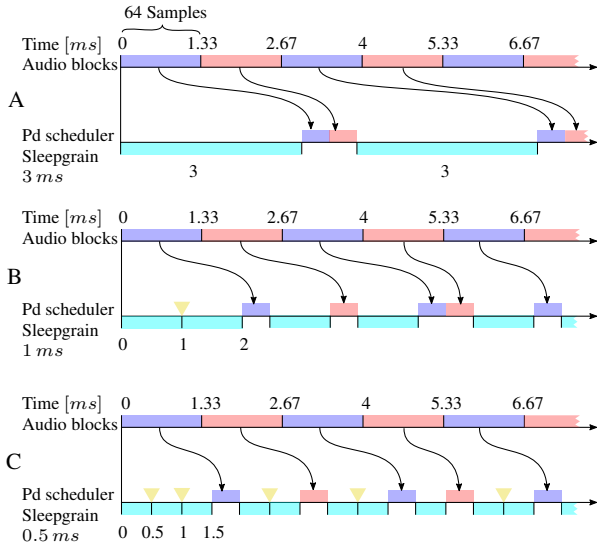


Figure 4. Processing of audio blocks with the Pd scheduler set to different *sleepgrain* sizes. A: Large *sleepgrain* sizes results in audio blocks being processed in bursts. B: A *sleepgrain* size with a similar duration like the blocksize in time results in few overhead function calls but causes jitter. C: Smaller *sleepgrain* sizes generate many overhead function calls but delivers a steadier computation of audio blocks. Overhead function calls, Sleep.

Fig. 4 illustrates how the *sleepgrain* parameter generates an overhead in function calls for small sizes. Note that this overhead does not harm the audio process, since the sleep calls are only generated cyanonce all audio and messaging work is done. The sleep function calls show a significant loading effect, when Pd is mostly idle, but if we run an audio process with more computational load to begin with (as shown in Fig. 5) this parameter has less influence on the total computational load. We only see a slight increase for small *sleepgrain* sizes, since Pd spends less time in idle state where additional function calls are generated. Even though DSP load increases for smaller *sleepgrain* sizes, buffer dropouts decrease. A side effect of this can be a sluggish or unresponsive graphical user interface, because background tasks will be slowed down.

To reduce overall system latency and allow the transfer of smaller audio blocks without bursting, we compiled Pure Data with an audio block size of 16 samples. This comes with the trade-off of processing overhead through function calls.

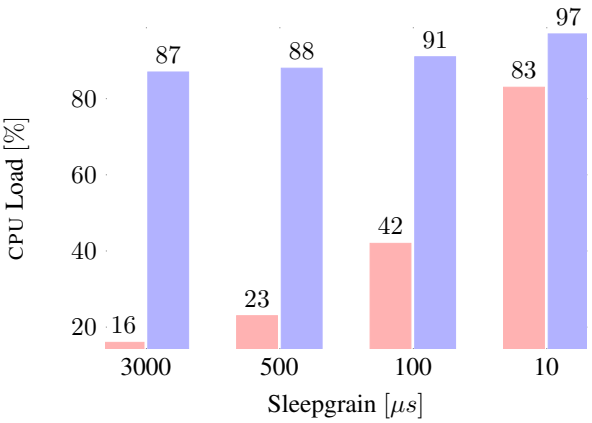


Figure 5. Low DSP load, High DSP load, The shorter the sleep time in Pd, the more virtual load will be on the CPU. The CPU load was measured with the in-built meter in Pd, averaged over 10 seconds.

5. JITTER/LATENCY MEASUREMENT SCHEMES

The total input to output latency measurements were taken using a square wave generator and an oscilloscope. Using two oscilloscope channels, the time difference between the original signal and the one sent through the device was measured. The period of the square wave was chosen to be longer than the latency of the system.

To quantify the block interval jitter we instantiated a counter in digital logic. This counter was driven by a low jitter 50 MHz clock source from external hardware. Each time a block of audio was written to the digital logic, the counter was reset and its value reported to a Pd external. This way we can get reliable timing information without needing to use system calls in our audio process functions.

Finally, we varied the block and *sleepgrain* sizes and calculated the block interval time from the counter value.

We took the minimum of the block interval time of 10,000 measurements for each setting and used this smallest interval to estimate the maximum use of capacity for the FPGA as follows:

[18] We found the sleepgrain size by calling `get_sys_sleepgrain()` in the sleepgrain external by IOhannes m zmölnig `https://git.iem.at/pd/zexy`. This agrees with the global delay setting being 3 ms.

[19] See `http://msp.ucsd.edu/Pd_documentation/x3.htm#s4`.

$$\text{FPGA}_{\max} = \frac{\min\{I_1, I_2, \ldots, I_m\}}{I_{rt}} \cdot 100 \qquad (1)$$

where:

$\text{FPGA}_{\max}$ = maximum use of FPGA capacity [%]
I_m = measured block intervals, $m = 10000$
I_{rt} = block interval in real time.

This number tells us the maximum load time of the FPGA where the likelihood of dropouts is very low.

5.1 Results

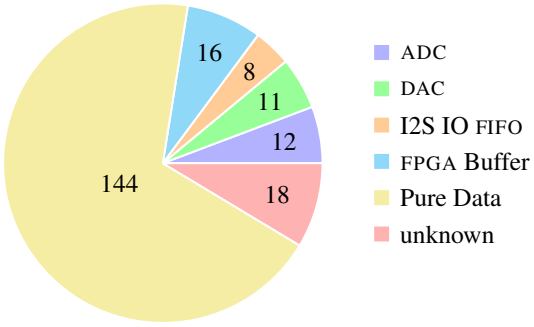


Figure 6. System latencies in samples. Total is 209 samples.

The total measured roundtrip latency was 4.36 ms. This accounts for all delays in the system: ADC and DAC conversions (23 samples), I2S sample buffer (8 samples), Pure Data's "Delay" setting (3 ms) and FPGA buffering (16 samples). This delay accounts for 209 samples at a sample rate of 48 kHz. Jitter was too small to measure with our measurement techniques and was lower than 0.1 ms. We did not investigate where the unknown additional latency of 18 samples arises from.

The share of delay added by the FPGA data transmission was 0.33 ms or 16 samples at 48 kHz.

Since we are using Pd on an embedded system where it is the process of highest priority, we can reduce the sleepgrain size to even 10 μs without major drawbacks. This system exhibits reasonable low latency for use as a digital musical instrument (4.36 ms) and leverages up to 88% of the theoretical processing power of its FPGA coprocessor (see Fig. 7).

6. CONCLUSIONS

We successfully demonstrated how to offload some of the complex audio processing to an FPGA coprocessor from an ARM processor running a Linux OS. From within a Pure Data application the FPGA computing is handled by an external. The latency is satisfactory, the system is stable (yet there is still room for further optimizations). A brief video can be found at `https://vimeo.com/668575690`.

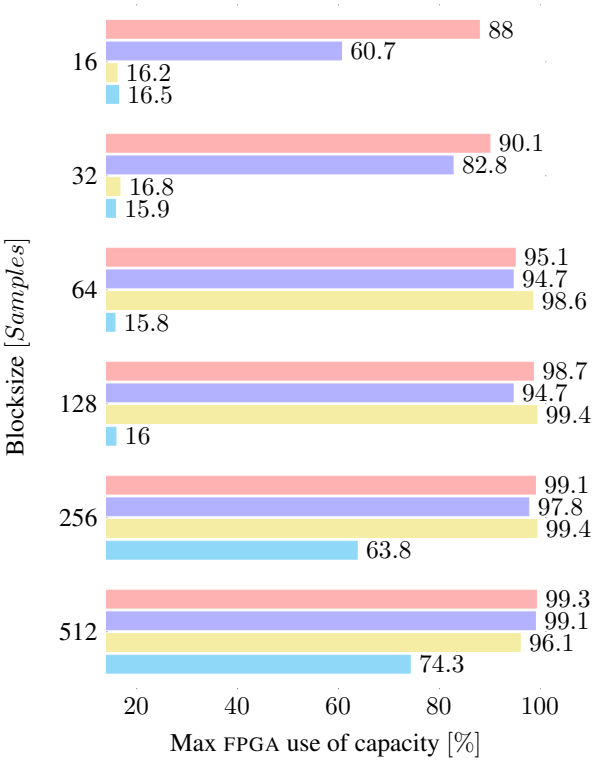


Figure 7. Sleepgrain size 10 μs, 100 μs, 500 μs, 3000 μs. The shorter the sleep time and the bigger the block size in Pd, the more compute time can be spent on the FPGA.

7. FUTURE WORK

- Allow multiple FPGA Pd-externals to coexist in one patch.
- Allow for uploading new bitstreams (the patterns for the cell fabric layout on the FPGA defining the DSP graph) to the FPGA from within Pd.
- In extension of the previous point: Let the user edit Verilog/VHDL from within Pd, compile on the fly and upload to FPGA. This would require open toolchains which are not available yet.
- Extend the simple mass-string models to more complex mass-interaction networks with non-linear interactions like collisions, penetrations, attraction, gravitation, parameters which are dynamic and/or conditional, etc.
- To further reduce latencies alternative hard- and software designs should be evaluated, such as replacing the Altera Linux kernel with a Xenomai 4 dual kernel or implementing the techniques found in BeagleRT [6].
- Speeding up Pd processing by using heavy compiler [20]. A benchmark must quantify the performance gain.
- Using the Faust to Verilog/VHDL compiling toolchain as in Risset et al. [12].

8. REFERENCES

[1] A. P. McPherson, R. H. Jack, and G. Moro, "Action-Sound Latency: Are Our Tools Fast Enough?" in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Brisbane, Jul. 2016, p. 6.

[2] J.-F. Charles, C. C. Solares, C. T. Tobon, and A. Willette, "Using the Axoloti Embedded Sound Processing Platform to Foster Experimentation and Creativity," in *Proceedings of NIME 2018*, Blacksburg, 2018, p. 2.

[3] R. Michon, Y. Orlarey, S. Letz, and D. Fober, "Real Time Audio Digital Signal Processing With Faust and the Teensy," in *Proceedings of the 16th Sound & Music Computing Conference*, Malaga, May 2019, p. 7.

[4] F. Reghenzani, G. Massari, and W. Fornaciari, "The Real-Time Linux Kernel: A Survey on PREEMPT_rt," *ACM Computing Surveys*, vol. 52, pp. 1–36, Jan. 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3297714

[5] E. Berdahl and W. Ju, "Satellite CCRMA: A Musical Interaction and Sound Synthesis Platform." in *NIME*, 2011, pp. 173–178.

[6] A. McPherson and V. Zappi, "An Environment for Submillisecond-Latency Audio and Sensor Processing on BeagleBone Black," in *Audio Engineering Society Convention 138*, Warsaw, May 2015. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=17755

[7] L. Turchet and C. Fischione, "Elk Audio OS: An Open Source Operating System for the Internet of Musical Things," *ACM Transactions on Internet of Things*, vol. 2, p. 18, Mar. 2021.

[8] L. Vignati, S. Zambon, and L. Turchet, "A Comparison of Real-Time Linux-Based Architectures for Embedded Musical Applications," *Journal of the Audio Engineering Society*, vol. 70, pp. 83–93, Jan. 2022, publisher: Audio Engineering Society. [Online]. Available: https://www.aes.org/e-lib/browse.cfm?elib=21553

[9] F. Pfeifle and R. Bader, "Real-time Finite Difference Physical Models of Musical Instruments on a Field Programmable Gate Array (FPGA)," in *Proc. of the 15th Int. Conference on Digital Audio Effects*, York, UK, 2012, p. 8.

[10] M. Verstraelen, F. Pfeifle, and R. Bader, "Feasibility analysis of real-time physical modeling using WaveCore processor technology on FPGA," in *Proceedings of the Third Vienna Talk on Music Acoustics*, University of Music and Performing Arts Vienna, Sep. 2015.

[11] T. Vannoy, T. B. Davis, C. Dack, D. Sobrero, and R. K. Snider, "An Open Audio Processing Platform using SoC FPGAs and Model-Based Development," in *New York*, 2019, p. 8.

[12] T. Risset, R. Michon, Y. Orlarey, S. Letz, G. Müller, and A. Gbadamosi, "Faust2FPGA for Ultra-Low Audio Latency: Preliminary work in the Syfala project," in *Proceedings of the 2nd International Faust Conference*, Paris, 2020, p. 10. [Online]. Available: https://ccrma.stanford.edu/~rmichon/publications/doc/ifc-20-faust2fpga.pdf

[13] M. Verstraelen, J. Kuper, and G. J. M. Smit, "Declaratively programmable Ultra Low-Latency Audio Effects Processing on FPGA," in *Proc. of the 17th Int. Conference on Digital Audio Effects*, Erlangen, Sep. 2014, p. 8.

[14] M. Neupert and C. Wegener, "Interacting with digital resonators by acoustic excitation," in *Proceedings of the 16th Sound & Music Computing Conference*. Malaga: Universidad de Málaga, May 2019, pp. 80–81. [Online]. Available: http://smc2019.uma.es/articles/D1/D1_01_SMC2019_paper.pdf

[15] V. Zappi, A. Allen, and S. Fels, "Shader-based Physical Modelling for the Design of Massive Digital Musical Instruments," in *Proceedings of the 2017 International Conference on New Interfaces for Musical Expression*, Aalborg, May 2017, p. 6.

[16] R. F. Molanes, J. J. Rodriguez-Andina, and J. Farina, "Performance Characterization and Design Guidelines for Efficient Processor–FPGA Communication in Cyclone V FPSoCs," *IEEE Transactions on Industrial Electronics*, vol. 65, pp. 4368–4377, May 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8082548/

[17] R. F. Molanes, L. Costas, J. J. Rodriguez-Andina, and J. Farina, "Comparative Analysis of Processor-FPGA Communication Performance in Low-Cost FPSoCs," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 3826–3835, Jun. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9165225/

[18] M. Puckette, *The theory and technique of electronic music*. Hackensack, NJ: World Scientific Publishing Co, 2007, oCLC: ocn166265250.

[19] J. Leonard, J. Villeneuve, R. Michon, Y. Orlarey, S. Letz, A. Three, and A. Four, "Formalizing Mass-Interaction Physical Modeling in Faust," *Stanford University*, p. 7, 2019.

[20] G. Moro, A. Bin, R. H. Jack, C. Heinrichs, and A. P. McPherson, "Making High-Performance Embedded Instruments with Bela and Pure Data," in *Proceedings of ICLI*, University of Sussex, Brighton, UK, Jun. 2016, p. 5.

DESIGNING INTERACTIVE SONIFICATIONS FOR THE EXPLORATION OF DANCE PHRASE EDGES

Andreas Bergsland
Norwegian University of Science and Technology, NTNU
andreas.bergsland@ntnu.no

ABSTRACT

The paper presents practice-led research where interactive sonifications of dance movements are created so as to have special emphasis on the movement edges of dance phrases. The paper starts out by discussing to what degree interactive sonification of movement and the artistic practice of interactive dance have areas of overlap, and whether their contexts and intentions may be combined to complement each other. A discussion of the concept of salience and how it relates to perceptual edges in general follows, and more specifically to onsets and endings both of movements and musical objects. In this discussion, the author also considers the role of accents as well as different degrees of abruptness of changes. The paper subsequently presents a set of interactive sonifications demonstrating different degrees of salience from very high to very low. Details related to the technical setup, analysis of movement data and the movement-sound mappings are presented, and the results are discussed.

1. INTRODUCTION

In the literature on interactive dance and sonification of movement, there are several mentions of the use of stillness and movement as productive or interesting oppositions [1–5]. For example, in pieces like *Seine hohle form* by Palindrome Dance Company, one can observe how in some sections dance movements are interspersed with stillness to create temporal incisions and accents. [1] In many of the phrase onsets and endings that occur, one can also observe a variation in temporal dynamics, with phrases beginning or ending gradually, abruptly or something in between, with or without accents. This has generated an interest in this author in exploring the dance phrase edges in the interactive sonification of dance movements through a practice-led process, also producing an artistic outcome.

In this paper, I would like to present some aspects of this work, including technical and perceptual issues as well as mapping, and how they relate to relevant research literature. It is not my aim to do a formalized evaluation of the sonifications in this context, but rather to discuss the grounding of core aspects of the sonifications in relevant scientific fields. To do this, I will start by looking at the field of interactive sonifications of body movements, and how it relates to the field of interactive dance before I turn to the concept of salience and discuss how different aspects of this concept can be relevant for the work presented in the paper.

[1] https://vimeo.com/8895552

Copyright: © 2022 Andreas Bergsland. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. BACKGROUND AND RELEVANT WORK

2.1 Interactive Sonification

In *The Sonification Handbook*, Walker and Nees define sonification as the data-dependent generation of sound where the transformation from data to sound is systematic, objective, and reproducible, and where the data relationships are comprehensible for a human listener [6]. Within this field, the sub-discipline of *interactive* sonification narrows the focus down to situations where humans interact with a system that transforms data into sound [7]. Thus, there seems to be a partial overlap between musical instruments and interactive sonifications, where the main difference is whether the perspective and function revolve around data or information and the conveying of that to a receiver, or whether it evolves around an artistic expression. A further sub-branch of interactive sonification is concerned with human movement as the source of data/information, a branch that Giomi labels *somatic sonification* [8]. Interestingly, he discusses the overlaps and distinguishing features between this branch of sonification and interactive dance, where his main distinguishing features are that sonification "provides information" with a goal of "understanding, exploring, interpreting, communicating or reasoning", whereas in an artistic performance "expressive purposes" and creating an "aesthetic experience" are paramount.

Although the field of sonification thereby seems to be distanced from artistic expressions such as music, it is interesting that aesthetics of sonification is also a part of the discussions in the field. Barrass and Vickers delineate a grey area between art and sonification, where if some data rooted in an extra-musical world in a musical composition is made explicit and the understanding of this can be aided by listening, this can qualify as a sonification [9]. Furthermore, they point to numerous examples from within the field where aesthetic perspectives are not in conflict with the intention of communicating information, but rather how these perspectives could be supporting communication.

2.2 Interactive Dance and Sonification

Interactive dance is a branch of artistic expression often considered to be conceived when John Cage and collaborators made the performance *Variations V* in 1965. Miller, in her in-depth study of this event, writes how with *Variations V* an art form was conceived where "dancers functioned as co-composers, exerting as much influence over the sonic landscape as the musicians who operated the electronic equipment", and where "the interaction of sound and motion was facilitated by a sophisticated technological component" [10]. Covering the more recent practices in the field, interactive dance has been defined as a "performance [...] in which a dancer's movement, gesture, and action are read by sensory devices, translated into digital information, processed by a computer program, and rendered into output that shapes the performance environment in real-time", and where, in the next stage, this output can affect the performers' actions [11]. This "output" might be varied, such as sound, video projections, lights, text, graphics, and robotic movement, but naturally it is the cases where sound is the output that is of interest here. Although the terminology is slightly different, we can recognize several similarities with interactive sonifications of body movements: bodily movements are represented by data (information) and translated (transformed) into sounding output.

If we return to interactive movement or somatic sonification it is interesting to note that Giomi and Leonard, even if maintaining the distinction between interactive dance as artistic practice and sonification as discussed above, present attempts at combining the two traditions [12]. More specifically, the authors present both others' and their own work, combining the aim for "aesthetically meaningful interactions" with the sonification paradigm and creating sounds that can reflect "objective properties of the movement". In many of these works the authors have aimed for different forms of sensorimotor learning and bodily awareness through sonic feedback, clearly in line with the sonification perspective. An even more radical approach is proposed by Barrass [13] who proposes to facilitate an "aesthetic turn" in sonification dissolving divisions between scientific and artistic methods based on the view that sound is "a naturally affective, aesthetic and cultural medium".

Taken together, interactive sonification of movements and interactive dance have overlapping principles and techniques for translating or transforming data from movements into sound, although there might be differences in the contexts (scientific vs. artistic), and main intentions (informative vs. aesthetic). These contexts and intentions might not be mutually exclusive, but may indeed be combined to complement each other. In the following section, I will present research related to the project's focus on temporal perceptual edges, more precisely dealing with how salience, accents, and segmentation can play a part in music/sound as well as dance, albeit in different ways.

2.3 Salience, Accents and Segmentation

Perceptual salience as a general phenomenon refers to the properties of an object or event that makes it stand out or pop out relative to what surrounds it and therefore likely will capture our attention [14, 15]. The concept is related to other phenomena such as scene analysis, segmentation, chunking, and Gestalt principles, and the key factor here is how perceptual contrasts and marked edges, which might be temporal and/or spatial, separate an entity from whatever surrounds it [16].

As for the auditory domain, qualitative temporal discontinuities in the signal, e.g. between sound and silence, are important, and a source for what Godøy labels *exogenous* (bottom-up) chunking [17]. Such discontinuities would e.g. appear if attacks (onsets) and endings were sufficiently abrupt. More gradual changes, on the other hand, would probably not be perceived as discontinuities [18]. It has to be noted, though, that segmentation/chunking can happen without demarcations of silence, but can happen both due to recognition of familiar musical units or schemata, such as a motif or a metric pattern. This is what Godøy calls *endogenous* (top-down) sources of chunking.

Another related type of high-salience temporal events in music are *accents* [19]. Accents can stand out from their surroundings due to several properties that distinguish them, as Thoresen notes, "most often by being louder; having a more ample mass, a brighter spectrum or a sharper onset quality; and/or being longer than other elements in their context" [20].

Since dance movements most often do not make a lot of sound, of course with several exceptions where body parts audibly touch each other, surfaces, or objects, the perception of another's dancing body is primarily visual. Although the same general principles of perceptual edges and contrast apply in the visual domain, the differences in modality imply a change in the spatio-temporal resolution, i.e. it is known that temporal acuteness is higher for auditory than for visual stimuli [21].

When it comes to the segmentation of dance phrases, top-down factors such as the familiarity of a genre, of specific dance movements, or the structure of repeated sections [22, 23], as well as bottom-up factors related to perceptual temporal edges can play a part. For the latter, analogous to the role of silence in musical segmentation, a period of stillness in between movements will also tend to indicate a boundary. For instance, Glowinski and colleagues found that subjects tend to be able to recognize pauses in a dance sequence when they are longer than 0.4s. of duration [24]. In comparison, most normal-hearing young adults will detect gaps of only 5 ms in a longer segment of white noise [25].

When perceiving dance and music together, one naturally needs to consider how the auditory and visual modalities work in tandem. Whereas the visual modality tends to dominate over the auditory modality in bimodal (audio-visual) spatial perception, the auditory modality tends to dominate over the visual modality in bimodal temporal perception [26]. Also, in experiments researchers have found that sound can 1. increase the salience of visual events, that is, if a sound is added to a visual event, it is more likely to attract attention [27], and 2. audition can affect how visual stimuli are perceived, especially when

sharp transient sounds coincide with the visual event [28]. This latter phenomenon is also referred to as auditory capture [29]. These findings might have an impact on the design of sonification of movements in an interactive dance setting.

3. PROCESS AND SETUP

3.1 Process

The work presented in this paper was carried out during a three-month guest researcher residency at the Zürich Hoch-schule der Künste (ZHdK). The residency culminated in a performance of interactive dance in November 2021 carried out as a collaboration between the author and dancer and choreographer Seh Yun Kim. The process leading towards the artistic performance was naturally influenced by the final artistic goal, and has affinities with Borgdorff's conception of artistic research by generating knowledge through an embodied and situated artistic practice embedded in artistic and academic contexts [30]. Moreover, the process has affinities with practice-led research and the author's role as "practitioner-researcher", as described by Grey: "The role is multifaceted - sometimes generator of the research material - art/design works, and participant in the creative process; sometimes self-observer through reflection on action and inaction, and through discussion with others; sometimes observer of others for placing the research in context, and gaining other perspectives; sometimes co-researcher, facilitator and research manager, especially of a collaborative project" [31].

The greater parts of the process, all the way up to the last rehearsals, also had many elements of an iterative design process [32]. As a rule, the work was organized into weekly cycles starting with a period of sound composition and technological development work by the author. This was then followed by a practice session, either in a dance studio or a performance lab with the author and the dancer together. Thus, the iterative work-cycle would typically involve 1) development of interactive instruments, 2) presentation and explanation of new or modified interactive instruments, 3) phase of free exploration with observation and taking notes, 4) reflecting together with dancer and taking notes, and finally, 5) agreeing on the further development of the interactive instruments.

Questions discussed in the fourth point of this cycle were typically:

- What was the general experience of the interaction?
- How do the sounds make the dancer feel?
- How did the dancer's emotions affect her movements?
- Was anything missing, felt awkward or difficult with the interaction for the dancer?

Thus, these questions guided our ideas for how we wanted to develop the instruments further, and this again was fed into the start of the cycle.

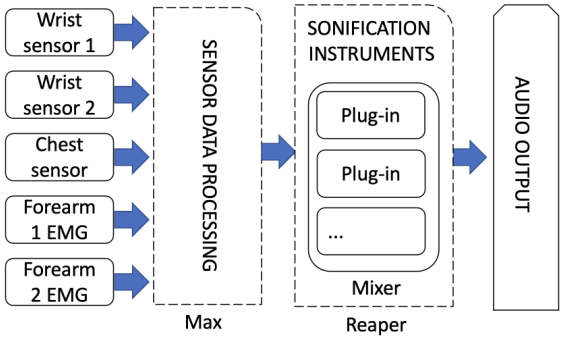


Figure 1. Technical setup used in the project including hardware and software elements.

3.2 Technical Setup and Sensor Placement

The technical setup in this project is shown in Figure 1. The data from the movement sensors was transferred to a computer via Wi-fi and Bluetooth and pre-processed in the Max software environment. The pre-processed data was transmitted over Open Sound Control (OSC) to a second computer running *Reaper*, a Digital Audio Workstation software. The sonifications were implemented as instruments in the sound and music computing system *Csound*, wrapped as VST-plugins with *Cabbage*, and inserted on separate tracks before sent to the audio output.

Several studies have shown how the choice of sensors and their placement on a dancer's body can affect the interaction possibilities and types of movement that an interactive dance system affords [33, 34]. Myo armbands [35] and NGIMU inertial sensors from X-IO [36] were chosen because they were 1) lightweight and relatively unobtrusive, 2) low-latency, 3) robust and reliable, 4) easy and fast to set up, and 5) did not restrict the dancer spatially. Furthermore, the Myo armbands were chosen due to their EMG sensing possibilities.

Although it was originally planned to have sensors on arms, legs, and torso, we ended up having sensors only on the arms and torso, mostly due to time constraints and the extra time implementing sensors and mappings for the legs would imply. This naturally led to a focus on the upper body throughout the process. The placement of the sensors can be seen in Figure 2. The placement of the NGIMUs on the dancer's wrists was seen as a good placement for an expressive use of the dancers' arms and hands, both regarding the choreographic and the musical aspects. The third NGIMU sensor was placed on the central upper chest. This placement was seen as interesting since it would allow for getting both torso direction and angle in a very straightforward way. The Myo armbands were placed on the upper part of the forearms, as devised by the manufacturer, to facilitate the tracking of finger and hand movements, along with the overall tension of the different forearm muscles.

4. MOVEMENT-SOUND MAPPINGS

During the project period and using the described setup, the author in collaboration with the dancer developed a

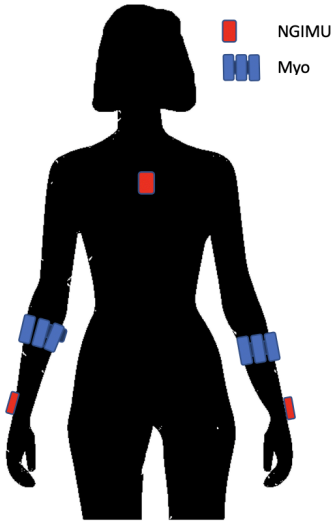


Figure 2. Placement of sensors on the dancer's body

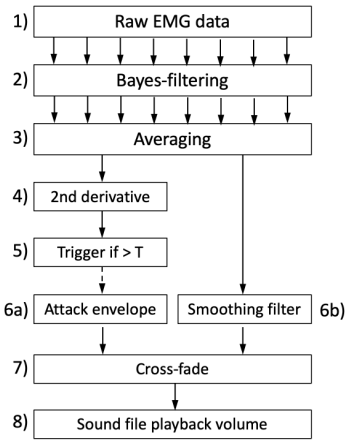


Figure 3. Mapping for clear onsets and smooth continuation

number of movement-sound mappings with special concern for the temporal edges of dance phrases. I will go through three of these in the following, focusing on the most relevant parameters in this context. Video examples of the mappings are provided as links in footnotes, and the reader is recommended to let these accompany the reading of the paper. [2]

4.1 Highly Salient Onsets with Smooth Continuation

The aim of this first mapping was to make abrupt onsets with sharp transients from small finger and hand movements, thereby aiming at high salience for the onsets of small movements. [3] Moreover, we wanted these to make it possible to let a sustained portion of the sound follow these onsets smoothly. In the context of the interactive dance performance, this was seen as a sort of introduction to the interactive paradigm and aimed at making the audience understand that the dancer is actually controlling the sounds, something which is considered important by several practitioners in the field [37].

Figure 3 shows the signal processing steps to achieve a mapping with very abrupt onsets and smooth continuation. The raw EMG data (1) was filtered with a Bayes filter from the *MuBu* package in *Max* (2). The filtered values were then averaged (3). The second derivative (4) was compared to a threshold, which created a trigger if the value was above it (5) that activated an attack envelope (6a). This envelope was cross-faded (7) with the smoothed EMG signal (6b) having a gradual onset. Using this cross-faded value to scale the volume of a sound file playback engine (8) with an arbitrary controllable time pointer made it possible to prolong the sound files indefinitely as long as the muscle activity was held above a certain threshold. [4] When the value dropped below this threshold, a relatively brief fade-out envelope would be imposed on the sound resulting in a relatively instant, albeit not marked, ending of the sound. It was a key point that the sound files had to have sharp transients at the beginning, and therefore clearly articulated stop consonants ([t], [k]) were recorded and used as sound material.

Several movement-sound mappings not directly related to onsets or endings were also applied in this interactive instrument more out of aesthetic concerns:

- Rotation of arm decides the sound file
- Muscle activation moves time pointer through sound
- Muscle activation modulates pitch
- Moving arm gives "chorus" effect
- Long sustained sound with stable pitch generates drone

All in all, this mapping made it possible to have a high-salience onset even from a tiny movement such as moving a single finger, combined with a smooth continuation. This also implies that the joint audiovisual salience of the movement onsets would be considerably increased, giving a heightened focus towards small movements that could otherwise seem to have little significance. In addition, several aspects of the sustained part of the sounds would follow the muscle activation in a way that had the potential to create variation and interest both for the performer and spectators.

4.2 Low and High Extremes of Salience

The second mapping had dual aims. [5] Firstly, to enable close to imperceptible onsets and endings, partly as a sort

[2] A video documenting the whole performance where these interactive sonifications were applied is found here: https://youtu.be/fkOZaT2pS-k

[3] A video demonstrating this sonification can be seen at https://youtu.be/BjW4027YKko.

[4] The playback engine, in this case, was the phase vocoder-based Csound opcode *mincer* which allows independent control of the time pointer that reads the file. For the attack portion of the sound, the time pointer would follow the timing of the original sound file. For the sustained portion, however, the time pointer would be going back and forth in the sustained portion of the sound, with the muscle activity directly controlling the pointer.

[5] A video demonstrating the sonification can be seen at https://youtu.be/Y0FabW2zQNk.

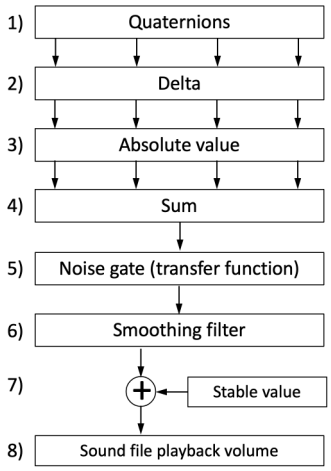


Figure 4. Mapping for gradual onsets and endings

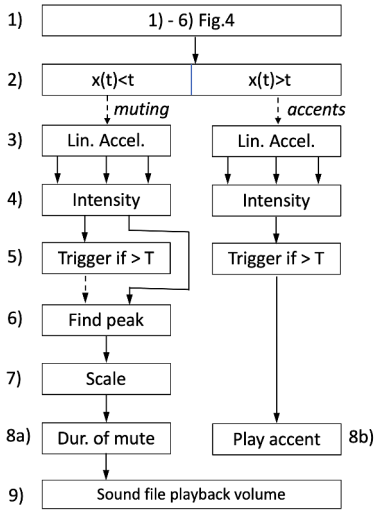


Figure 5. Mapping for alternative endings

of antithesis to the first mapping. Secondly, to highlight high-salience movements with a special effect.

A central strategy for the first aim was to introduce a "carpet" of sound that was always present independent of any movement, and then construct the mapping so very gentle arm movements would bring this carpet gradually to the foreground. In the same manner, as the first mapping, vocal sound recordings were used, but this time with very gradual onsets, and also with three different layers giving a hint of a choral quality, which also added richness to the sound. A playback engine with time pointer control made it possible to prolong the sound indefinitely. [6] For the second aim, a computational model of saliency called the *rarity index* was done based on Niewiadomski and colleagues with a time window of 2.5 seconds [38]. The algorithm was slightly modified to only give high output values when there was a large positive change in input magnitude, i.e. from zero or a lower value to a higher value, and not the other way around.

The mapping for the gradual onset and ending part of this sonification is seen in Figure 4. A general movement intensity or activity value was calculated using the absolute value (3) of the delta (2) of the quaternions (1) from the two NGIMU sensors on the dancer's wrists. These values were summed (4) before being treated with a simple noise gate (5), implemented as a linear transfer function set to 0 up to the noise threshold. This activity value was sensitive even for very low-velocity movements. By adding this to a small constant (7) which was used to scale the volume of the sound file (8), the sound would always be playing. This, combined with the sensitivity to slow movements, made it possible to achieve close very soft, and gradual onsets and endings of the sound, thus exploring the minimal range of salience.

In the second layer of the mapping which was sensitive to high-salience movement events, the rarity index value was mapped to a pitch transposition and spectral arpeggiation effect, including the volume of the effect. This mapping worked as a highly expressive means for the dancer when going from relative stillness to the upper intensity range of her movements. When the dancer produced a sudden burst of activity with her arm, the sounding result was almost like a short phrase played on a flute with a marked timbral and dynamic contrast compared to the relatively quiet vocal layers.

As for the previous mapping, several features were also added with concern for the aesthetic qualities of the sound:

- Brief period of stillness changes one of three sound layers
- Arm horizontal orientation guides time pointer in the sound file
- Salient movements bring forth a pitch modulation and spectral arpeggiation effect

In sum, this sonification explored the high and low ends of movement salience, combining a "carpet" of vocal sounds for the very slow and gradual movements, and more dramatic spectral effects for the highly salient movements.

4.3 High Salience Endings

The third mapping discussed in this paper focused on achieving a rich, complex and varied sonification of dance phrases with two types of high-salience endings. [7] Since the endings are of most relevance here, I will start by describing these in detail, and get back to a few more general traits below. The first type of ending was achieved with a sudden muting of the sound, whereas the second was achieved using an accent sound of the impulse-resonance type according to Smalley's spectromorphological typologies [39].

A simplified flowchart for this mapping is shown in Figure 5. At the outset a calculation based on delta quaternions (1), as shown in points 1-6 of Figure 4 is then sent to

[6] This time a granular playback engine was chosen, the Csound opcode *partikkel*, which gave a slightly more quavering quality to the sound.

[7] A demonstration of the sonification can be found at https://youtu.be/3yd6XDe7zVU.

a switch (2), which will compare it to a threshold value and decide whether to mute or make an accent at the ending of the sound: it will proceed with the left part of the flowchart if lower than the threshold, or the right if higher than the threshold.

The first three steps (3)-(5), are similar for both paths: The linear acceleration (3) values from the NGIMU sensors on the wrists are sent to the *intensity* object in the Max software. [8] If this value is above a threshold, it will produce a trigger according to the chosen path. For the muting path, the peak intensity (6) will be scaled (7), and used to set the duration of the muting (8a) before being applied as an envelope of the sound file (9). For the accents path, it will simply trigger a sound file with an accent sound.

As for the general design of this sonification, it is a form of granular concatenative synthesis based on the author's earlier work [2]. The sonification uses a great number (200+) of short sound files, in this case, recordings of different metal objects, organized to make up a perceptual continuum. These sound files are triggered by a metronome whose frequency is controlled by the delta quaternion-based intensity parameter (implemented as in 4, pt.(1)-(6)). The horizontal orientation of the NGIMU-sensor on the torso will then select which sound file is played back. The result is a rich and complex sonification with expressive potential.

Taken together, it has to be admitted that the aim of designing endings with high salience was only partly successful for this sonification. This might have to do with the high degree of variation and timbral richness in the general design of the sonification, which somehow slightly reduced the effect of the sudden muting and the accent. It might also be that for the accents, the impulse-resonance sounds that were chosen could have been even more salient without the resonance part, thereby articulating the contrast to the following silence in a temporally more acute way. Moreover, the technical implementation relied on an increase of intensity of an ending gesture, but such an increase of intensity can also happen without the movement ending, thus producing muting or accents without any actual movement phrase ending. Therefore, to be effective, this sonification demands some training to get the desired effects at the right moments and consistency on the part of the dancer in using it for movement endings.

5. DISCUSSION

The three sonifications presented in this paper were all originally developed for an artistic project, implying that compositional ideas, structures, and developments through the different sections of the work made sense from an aesthetic point of view. Nevertheless, the focus on beginnings and endings, especially regarding salience as perceptual edges, segmentation, and accents involving both movement, music, and sound, has been an important backdrop for the work.

[8] The algorithm is a part of the RIoT Bitalino package and is based on taking the delta of each of the values, squaring them, smoothing the result with a first-order IIR filter, scaling, and finally summing the values.

The sonifications in this project have suggested different ways of giving high salience to movement edges, thereby both providing heightened temporal resolution to these movements through the auditory modality and potential of putting a special focus on aspects of a movement that in other settings would have been lost.

The role of salient temporal edges for segmentation can also prove useful. Making onset and ending points salient may assist chunking operation and thereby create a clear phrase structure. This can probably be of assistance in movement learning and memorization, whereas in other cases a familiarity with genre and repertoire is needed [22]. Using some of the techniques of sonification presented in this paper, one could e.g. speculate that dance phrases could have been segmented with a lot shorter pauses than the 0.4 seconds that Glowinski and colleagues found in their study [24]. Using sonifications with high salience edges could also be important if one wishes to set movement phrases in relation to other temporal phenomena, synchronize, create a rhythmical temporal response, etc.

There can also be motivations for the design of interaction with low temporal salience, as presented in the second sonification here with the always present vocal layers. Fleeting experiences with few or no temporal incisions can create a high orientation towards the present, or an "out-of-time" experience, which e.g. can be paramount to let the user relax or in a state of flow that can be useful in therapy, rehabilitation, and more.

Lastly, for all the sonifications in this paper, aesthetic issues have been crucial in the design. Creating variation in the sound material while retaining a sense of coherence and continuum, has been one of the core design principles of this author. This has included the use of recorded sound samples, often in large numbers, organized in perceptual continua. It is the author's experience that this has resulted in interactive sonifications that have created engagement, playfulness, and enjoyment for the user and avoided annoyance and irritation. While sound sources like sine tones and white noise are easily reproducible and might represent certain types of data more objectively, their uniformity can imply a risk of annoyance and listener fatigue, which variation can counteract [9].

6. CONCLUSIONS AND OUTLOOK

During this paper, I hope I have shown that set of movement-sound mappings presented in this paper are sonifications based on providing information about the temporal dynamics of the movements, with special emphasis on the onsets and endings. While they have had many aspects that were designed with an aesthetic motivation, this might accommodate the user in receiving the information, simply by making the sonification more engaging to use and less prone to annoyance. The precise mechanisms involved will still have to be verified by an empirical evaluation in the future. Moreover, while several of the mappings presented seemed to have clear high-salience as well as low-salience events, further details of the landscape in between remain to be explored.

Acknowledgments

This research was funded by the Norwegian University of Science and Technology (NTNU). Thanks to Anita Schuster, Leonie Wanger, Michael Schwarze, Naemi Haab, Elsbeth Legler Thomsen and Vera Rieger for providing vocal material for the sonifications. Furthermore, thanks to Simon Kötz and Tobias Gerber for help with sound recording and to Peter Färber for technical assistance. Also many thanks to the ICST(ZHdK) and Casa Paganini (DIBRIS, University of Genova) for access to resources and facilities during different phases of this work.

7. REFERENCES

[1] S. F. Alaoui, F. Bevilacqua, B. B. Pascual, and C. Jacquemin, "Dance interaction with physical model visuals based on movement qualities," *International Journal of Arts and Technology*, vol. 6, no. 4, pp. 357–387, 2013.

[2] A. Bergsland and R. Wechsler, "Composing interactive dance pieces for the MotionComposer, a device for persons with disabilities," in *Proceedings of the International Conference on New Interfaces for Musical Expression NIME'15*, E. Berdahl, Ed. Lousiana State University, 2015, Conference Paper, pp. 20–24.

[3] J. Birringer, "Dance and interactivity," *Dance Research Journal*, vol. 35/36, pp. 88–112, 2004. [Online]. Available: www.jstor.org/stable/30045071

[4] C. Erdem, K. H. Schia, and A. R. Jensenius, "Vrengt: A shared body–machine instrument for music–dance performance," in *Proceedings of the International Conference on New Interfaces for Musical Expression, NIME'19*. Universidade Federal do Rio Grande do Sul, 2019, Conference Paper, pp. 186–191.

[5] R. Masu, N. N. Correia, S. Jurgens, J. Feitsch, and T. Romão, "Designing interactive sonic artefacts for dance performance: an ecological approach," in *AM'20: Audio Mostly*, 2020, Conference Paper, pp. 445–129.

[6] B. N. Walker and M. A. Nees, *Theory of Sonification*. Berlin, Germany: Logos Verlag, 2011, book section 2, pp. 9–39.

[7] J. Yang, T. Hermann, and R. Bresin, "Introduction to the special issue on interactive sonification," *Journal on Multimodal User Interfaces*, vol. 13, no. 3, pp. 151–153, 2019. [Online]. Available: https://doi.org/10.1007/s12193-019-00312-z

[8] A. Giomi, "Somatic sonification in dance performances. from the artistic to the perceptual and back," in *Proceedings of the 7th International Conference on Movement and Computing, MOCO'20*, 2020, Conference Paper, pp. 1–8.

[9] S. Barrass and P. Wickers, *Sonification Design and Aesthetics*. Berlin, Germany: Logos Verlag, 2011, pp. 145–172.

[10] L. E. Miller, "Cage, Cunningham, and collaborators: The odyssey of 'Variations V'," *The Musical Quarterly*, vol. 85, no. 3, pp. 545–567, 2001. [Online]. Available: http://www.jstor.org/stable/3600996

[11] E. Mullis, "Dance, interactive technology, and the device paradigm," *Dance Research Journal*, vol. 45, no. 03, pp. 111–123, 2013. [Online]. Available: http://dx.doi.org/10.1017/S0149767712000290

[12] A. Giomi and J. Leonard, "Towards an interactive model-based sonification of hand gesture for dance performance," in *Proceedings of the International Conference on New Interfaces for Musical Expression, NIME'20*, 2020, Conference Paper, pp. 369–374.

[13] S. Barrass, "The aesthetic turn in sonification towards a social and cultural medium," *AI & SOCIETY*, vol. 27, no. 2, pp. 177–181, 2012. [Online]. Available: https://doi.org/10.1007/s00146-011-0335-5

[14] D. Kerzel and J. Schönhammer, "Salient stimuli capture attention and action," *Attention, Perception, & Psychophysics*, vol. 75, no. 8, pp. 1633–1643, 2013. [Online]. Available: https://doi.org/10.3758/s13414-013-0512-3

[15] S. Treue, "Visual attention: the where, what, how and why of saliency," *Current Opinion in Neurobiology*, vol. 13, no. 4, pp. 428–432, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959438803001053

[16] J. C. Schacher and D. Bisig, "Watch this! expressive movement in electronic music performance," in *Proceedings of the 2014 International Workshop on Movement and Computing*, 2014, Conference Paper, pp. 106–111.

[17] R. I. Godøy, A. R. Jensenius, and K. Nymoen, "Chunking in music by coarticulation," *Acta Acustica united with Acustica*, vol. 96, no. 4, pp. 690–700, 2010.

[18] A. Bergsland, "Experiencing voices in electroacoustic music," PhD thesis, Norwegian University of Science and Tehcnology, 2010.

[19] C. Drake and C. Palmer, "Accent structures in music performance," *Music perception*, vol. 10, no. 3, pp. 343–378, 1993.

[20] L. Thoresen, "Form-building patterns and metaphorical meaning," *Organised Sound*, vol. 15, no. 02, pp. 82–95, 2010. [Online]. Available: http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=7829196&fileId=S1355771810000075

[21] T. H. Rammsayer, N. Borter, and S. J. Troche, "Visual-auditory differences in duration discrimination of intervals in the subsecond and second range," *Frontiers in Psychology*, vol. 6, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpsyg.2015.01626

[22] B. E. Bläsing, "Segmentation of dance movement: effects of expertise, visual familiarity, motor experience and music," *Frontiers in Psychology*, vol. 5, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpsyg.2014.01500

[23] V. M. Dyaberi, H. Sundaram, J. James, and G. Qian, "Phrase structure detection in dance," in *Proceedings of the 12th Annual ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2004, Conference Paper, p. 332–335. [Online]. Available: https://doi.org/10.1145/1027527.1027604

[24] D. Glowinski, A. Camurri, C. Chiorri, B. Mazzarino, and G. Volpe, "Validation of an algorithm for segmentation of full-body movement sequences by perception: A pilot experiment," in *International Gesture Workshop*. Springer, 2007, Conference Paper, pp. 239–244.

[25] A. Giannela Samelli and E. Schochat, "The gaps-in-noise test: Gap detection thresholds in normal-hearing young adults," *International Journal of Audiology*, vol. 47, no. 5, pp. 238–245, 2008, doi: 10.1080/14992020801908244. [Online]. Available: https://doi.org/10.1080/14992020801908244

[26] L. Ortega, E. Guzman-Martinez, M. Grabowecky, and S. Suzuki, "Audition dominates vision in duration perception irrespective of salience, attention, and temporal discriminability," *Attention, Perception, & Psychophysics*, vol. 76, no. 5, pp. 1485–1502, 2014. [Online]. Available: https://doi.org/10.3758/s13414-014-0663-x

[27] T. Noesselt, D. Bergmann, M. Hake, H.-J. Heinze, and R. Fendrich, "Sound increases the saliency of visual events," *Brain research*, vol. 1220, pp. 157–163, 2008.

[28] S. Shimojo, C. Scheier, R. Nijhawan, L. Shams, Y. Kamitani, and K. Watanabe, "Beyond perceptual modality: Auditory effects on visual perception," *Acoustical Science and Technology*, vol. 22, no. 2, pp. 61–67, 2001.

[29] S. Morein-Zamir, S. Soto-Faraco, and A. Kingstone, "Auditory capture of vision: examining temporal ventriloquism," *Cognitive Brain Research*, vol. 17, no. 1, pp. 154–163, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926641003000892

[30] H. Borgdorff, *The Production of Knowledge in Artistic Research*. London: Routledge, 2011, book section 3, pp. 44–63.

[31] C. Grey, *Inquiry through practice: developing appropriate research strategies*. Helsinki: Research Institute, University of Art and Design, Helsinki UIAH, 1998, pp. 82–95.

[32] A. Pratt, *Interactive design : an introduction to the theory and application of user-centered design*. Beverly, MA: Rockport, 2012.

[33] A. Bergsland, S. Saue, and P. Stokke, "VIBRA-technical and artistic issues in an interactive dance project," in *16th Sound and Music Computing, SMC'19*, I. Barbancho, L. J. Tardón, A. Peinado, and A. M. Barbancho, Eds., 2019, Conference Paper, pp. 39–46.

[34] T. Svenns, "SENSITIV: Designing for interactive dance and the experience of control," Master's thesis, KTH, 2020. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1466897/FULLTEXT01.pdf

[35] K. Nymoen, M. R. Haugen, and A. R. Jensenius, "Mumyo – evaluating and exploring the myo armband for musical interaction," in *Proceedings of The International Conference on New Interfaces of Musical Expression Conference*, 2015, Conference Paper, pp. 215–219.

[36] "Ngimu - x-io technologies," February 2022. [Online]. Available: https://x-io.co.uk/ngimu/

[37] J. Toenjes, "Composing for interactive dance: Paradigms for perception," *Perspectives of New Music*, vol. 45, no. 2, pp. 28–50, 2007, empowerment, blurring of roles,. [Online]. Available: http://www.jstor.org/stable/25164655

[38] R. Niewiadomski, M. Mancini, A. Cera, S. Piana, C. Canepa, and A. Camurri, "Does embodied training improve the recognition of mid-level expressive movement qualities sonification?" *Journal on Multimodal User Interfaces*, vol. 13, no. 3, pp. 191–203, 2019.

[39] D. Smalley, "Spectromorphology: explaining sound-shapes," *Organised Sound*, vol. 2, no. 2, pp. 107–126, 1997.

Effect of Using Individually Simulated HRTFs on the Outcome of Tournament Selection Procedures in a Virtual Environment.

Michael Oehler **Minh Voong** **Marlon Regener** **Maurício do V. M. da Costa**

Music Technology & Digital Musicology Lab (MTDML),
Osnabrück University, Germany,
michael.oehler@uos.de

ABSTRACT

Individualized head related transfer functions (HRTF) play an increasingly important role in the field of virtual acoustics, especially for the perceived immersion in virtual environments. A central question in this context is often how exactly the individual fit is realized best. In the current exploratory study, a perception based tournament task is combined with numerically simulated HRTFs. HRTF selection performed in this way is relatively time-saving and does not require much technical effort. The two main objectives are to analyze the impact of numerically simulated individual HRTFs on the outcome of a tournament task and to investigate whether this correlates with the performance in a localization task in which the participants have to spatially localize sounds using the different HRTFs from the tournament task. The study is divided into 3 consecutive parts. First, the participants' individual HRTFs are simulated based on a 3D scan, then each participant takes part in the tournament task to identify the best fitting HRTF, and then the best (and worst) fitting HRTFs are validated in a localization task along with the simulated HRTF. The results of the tournament task show that the numerically simulated individual HRTFs had little effect on the outcome. In the localization task, however, the simulated HRTFs produced the best results, on average.

1. INTRODUCTION

Individualized head related transfer functions (HRTF) play an increasingly important role in the field of virtual acoustics, especially for the acoustic layer in virtual environments (VE). Various studies have shown that well-fitting HRTFs have a positive effect on the perceived immersion in virtual (VR), augmented (AR) or mixed (MR) reality applications [1]. This is especially relevant for music-related applications, where the acoustic domain is of particular importance. However, shortcomings in the determination of the individually appropriate HRTF may result in little or no advantage over generic HRTFs [2]. This depends on many factors, such as the specific measurement procedure [2], the type of validation [3], or the inclusion of additional sensory modalities [4].

A central question in this context is often how exactly the individual fit is realized best. A detailed overview of the different approaches to customizing HRTFs is provided in [5]. Acoustic measurements, the probably most obvious method, are usually very time and resource consuming, although new approaches have now been developed to rapidly measure HRTFs in ordinary home environments. For a recent review on different approaches in this area see [6]. Another technique is numerical simulation, e.g., using Fast-Multipole-accelerated Boundary Element Method (FM-BEM) [7], Finite Difference Time Domain Method (FDTD) [8] or Finite Element Method (FEM) [9]. Comparisons between numerical simulations and acoustic measurements have been around for a long time (e.g. [10]), but have been increasingly discussed again in recent years.

Besides the approach of indirect individualization based on anthropometric data [11-13], individualization based on perceptual feedback seems to be promising for practical everyday use, since HRTF selection performed in this way is relatively time-saving and possible without significant technical effort. Several selection methods have been tested for this purpose [14-17]; a good compromise between accuracy and time required is provided by the Swiss tournament system, for example [18,19]. In corresponding tournament tasks, participants are usually asked to evaluate the difference between two HRTFs based on certain perceptual criteria such as first impression, envelopment, and externalization.

In previous studies, we tested whether a corresponding tournament task could be used to identify individual HRTFs for each person who would also perform best in a spatial localization task [18, 20]. Although better results tended to be found in the localization task for the "winning" HRTFs of the tournament task, the effect sizes were rather small. This could be partly due to the fact that the pre-selection of the HRTFs in the tournament task was realized via a random selection from common HRTF libraries (CIPIC, LISTEN, etc.) and thus this specific selection may have been less well suited to individual participants.

Therefore, in the current study, a tournament task is developed including each participant's individually simulated HRTF in addition to a random selection of profiles from common databases. Although there are numerous sources of potential bias in the numerical simulation of HRTFs that distort the resulting HRTF compared to the participant's actual HRTF (e.g. when

creating/scanning 3D models or post-processing meshes), it is assumed that an HRTF that is not equal to, but close to, the participant's own HRTF will score better than the random HRTFs from the databases. Should this be confirmed, simulated HRTFs could then be included in the compilation of individual HRTF profiles of tournament sets in future studies, e.g., multiple simulated models of an individual that differ in certain parameters that, however, still need to be determined.

Accordingly, the two main objectives of the present study are (a) to analyze the impact of numerically simulated individual HRTFs on the outcome of a tournament task that uses perceptual categories such as externalization, envelopment, etc., and (b) to investigate whether this correlates with the performance in a localization task in which the participants have to spatially localize sounds using the different HRTFs from the tournament task.

Another general objective is to orient the methods used in the study towards a future application-oriented suitability for everyday use. This means, for example, that the scans of the 3D models for the numerical simulation should not be performed with expensive, difficult-to-access and usually permanently installed laboratory scanners, but with solutions comparable to a scan with (e.g., LIDAR) scanners of future smartphone generations. It is assumed that the loss of precision compared to laboratory scanners and conditions can then (partly) be compensated by the proposed tournament procedure with several variants of the individually simulated HRTF. However, this aspect goes beyond the scope of the current study.

Furthermore, the listening experiment is to be realized with bone conducting headphones (BCH) and not with "regular" headphones (RH) placed in, on or above the ear. Especially for MR or AR applications that rely on the perception of both natural and virtual sounds, the use of RH is less suitable [21]. Since BCH transmits sound via vibrations and the speakers are placed behind the ear or at the temple, i.e., they radiate sound directly into the inner ear [22], they are more suitable for transmitting the acoustic signal in AR or MR applications. Of course, this could be a critical factor that biases the results in both tasks and therefore needs to be carefully controlled. On the other hand, previous studies have shown that there is no significant difference between the use of RH and BCH in comparable tournament and localization tasks [20]. Although there are currently no commercial MR or AR products on the market that use BCH or comparable technologies (e.g., cartilage conduction), the potential benefits are obvious.

2. METHOD

The study is divided into 3 consecutive parts. First, the participants' individual HRTFs are simulated, then each participant takes part in the tournament task to identify the best fitting HRTF, and then the best fitting HRTF is validated in a localization task along with the HRTF ranked last and the simulated HRTF for comparison.

2.1 Participants

A total of 10 participants took part in the study (3 female, 7 male, M = 25.6, SD = 4.54). All participants had to go through both the tournament task and the localization task and all of them had normal hearing.

2.2 Numerical Simulation of HRTFs

The numerical simulations were based on the following phases: (a) 3D scanning, in which a scan of each participant is taken, resulting on a 3D mesh; (b) pre-processing, where the meshes are digitally treated and prepared for the numerical calculations; (c) numerical calculation, in which the transfer function between several points in space was computed for the faces indicated on the participant's mesh as left and right ears; and (d) post-processing, in which the results were converted into a single file containing the HRIRs for each participant. Such steps are detailed in the following.

2.2.1 3D Scanning

In order to get as close as possible to a future application-oriented suitability for everyday use, the 3D-scans of the participants were performed using the consumer-friendly and relatively affordable POP 3D Scanner by Revopoint[1]. Different approaches were tested to get the best possible scans with the available hardware. A free-roaming handheld scan offers the most promising results so far (which is encouraging for a scan provided by a smartphone will almost always be handheld). Further automation and routines for the scanning process are planned for future research. The participants were scanned from the utmost tip of the head to the top of the breast. Detailed scanning was required in the area of the ears while the remaining parts only needed to be roughly captured in order to get the data required for the HRTF simulation. The meshed models were then exported as Polygon File Format (*.ply) and partly edited in post-processing to smooth out rough textures and align the model in the right direction.

2.2.2 Numerical Calculation

Once the 3D meshes were generated, a pre-processing procedure was performed in Blender,[2] a free open-source software for 3D modeling. There, unwanted details were

Copyright: 2022 Michael Oehler et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any médium, provided the original author and source are credited.

[1] See https://de.shop.revopoint3d.com (last viewed: Mar. 30, 22)

[2] Available from http://www.blender.org (last viewed: Feb. 03, 22)

smoothed out and artifacts related to the scanning were corrected. Different resolutions were adopted for different regions of the mesh, as to reduce the computational burden. The ears were re-meshed to have a resolution of 1.3 mm, which seemed to be compatible with the details provided by the scanner; the region near the ears and the rest of the head had 3 mm and 6 mm resolutions, respectively; shoulders had 16 mm, and the upper section of the torso had a 33 mm resolution. Figure 1 depicts a participant's mesh, in which one can see the final result of the pre-processing procedure just described.

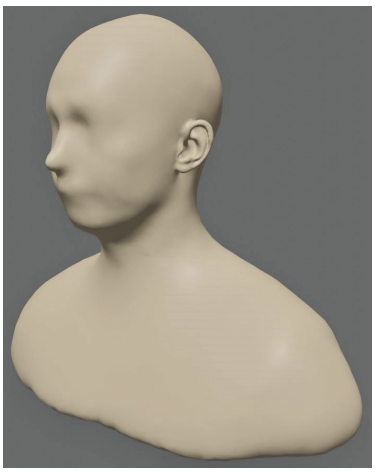


Figure 1. Snapshot of a participant's mesh after pre-processing.

After the meshes were pre-processed, the numerical calculations were conducted using the MESH2HRTF [23,24], which is an open-source library based on the fast-multipole BEM solver. The BEM simulations were performed for the frequency spectrum ranging from 100 Hz to 16 kHz with a 100 Hz step, and the solutions were sampled at 1550 collocated spatial directions distributed at the surface of a sphere with 1.2 m of radius centered at the participant's head. The final HRIRs were then resampled to a sampling rate of 44.1 kHz.

2.3 Tournament Task

A tournament procedure was chosen on the one hand to ensure comparability with previous studies, e.g. [18,20], and on the other hand because it allows several HRTFs to be included in the comparison at the same time in a relatively time-saving manner. The tournament task is mostly identical to the implementation in [18], where participants had to compare different HRTFs in a virtual acoustic environment. In the current study, the experiment itself was also conducted in a VE. In other words, participants can not only hear the sound sources, but also see them. The environment consists of the urban soundscape of a city park with different sound sources. The focus was on high ecological validity by adding objects with their corresponding sounds that fit into an urban park, e.g., car sounds, bird sounds, and a drone flying in a circle above the participant.

The Swiss tournament format, in which equal participants compete against each other, has proven useful. This format is widely used in chess and other sports and offers a good compromise between the time-consuming round robin format and a knock-out system, where promising HRTFs might be sorted out too early due to unforeseen circumstances [18].

Care was taken to ensure that subjects did not use one HRTF for a disproportionately long or short time in each pairwise comparison. For this purpose, the respective time spans were recorded and, in addition, irregularities were noted in the experimental protocol.

2.3.1 Apparatus

To ensure an immersive experience and allowing unrestricted exploration of the environment, the HTC VIVE Pro Eye [25] (including wireless adapter) was used for the study. The VR system's built-in on-ear headphones were not used, as the audio signal was played back through the bone-conducting TREKZ Titanium headphones[3] via Bluetooth. A VIVE controller is used as a virtual pointing tool to make judgments. The room size in which participants can move around is about nine square meters.

For the creation of the virtual environment, Unity [26] is used as a software framework. Its modularity is an important feature, as it makes it easy to implement plugins and additional features in existing projects. In this case, the SOFAlizer plugin [27] adds an auditory spatialization function to Unity's sound engine, which can be used to perform interpolations in relation to the distance of the measurement directions of the HRTF. Another feature is the option to preload up to ten user-defined HRTFs in memory and swap them on-the-fly. This makes it possible for the participants to perceive and evaluate the acoustic environment in real time with different HRTFs. On the other hand, audio signal processing features in Unity are limited to basic spatialization functions, as the focus of the Unity engine is more on visual components. Further processing such as phase shifting or other functions are not offered and rely heavily on third-party solutions.

2.3.2 Test Procedure

The participants are placed in the middle of a virtual urban city park scene, where they can move freely and are solely limited to the space in reality. A virtual display floating in the environment shows the study instructions and also allows the input of data during the study. After recording the general data, such as age and experience with VR environments, the tournament phase begins. Six HRTFs, mostly from previous studies [18,20] and proven to be suitable for localization tasks in several other studies [28,29], compete against each other. For this study the HRTF of the KEMAR artificial head [30] and the

[3] Discontinued product; actual product portfolio in https://shokz.com/ (date last viewed: Feb. 03, 22).

participant's own HRTF are included, increasing the total number of HRTFs to eight (see Tab. 1). The audio of the scene is enabled as soon as the task begins.

HRTF0	LISTEN_1014	Resolution Elevation	
HRTF1	CIPIC_058	Kemar	50 locations
HRTF2	LISTEN_1022	CIPIC	50 locations
HRTF3	KEMAR head	LISTEN	50 locations
HRTF4	LISTEN_1028	Resolution Azimuth	
HRTF5	Personal HRTF	Kemar	approx 5° increments
HRTF6	LISTEN_1049	CIPIC	approx 5° increments
HRTF7	CIPIC_124	LISTEN	15° increments
TestSignal		Measurement Distance	
Kemar	ML Sequence	Kemar	1.4m
CIPIC	Golay-Code	CIPIC	1m
LISTEN	Sweep	LISTEN	1.95m

Table 1. Used HRTFs in the tournament task [31-33].

The tournament consists of six rounds with four matches each. In each match two HRTFs are competing against each other. The participant is asked to rate the difference between these two HRTFs using different criteria, such as *first impression*, *envelopment*, and *externalization*, for three different sound sources: cars, birds, and the drone. Envelopment describes the degree to which the user feels acoustically embedded in the scene, while externalization indicates the degree to which the participant perceives the sound as an external source, e.g., the absence of voice-of-god-artifacts. The different increments on each slider range from 'A far better than B', 'A better than B' and 'A slightly better than B' to 'B far better than A' (see Fig. 2). During the match, the participant can switch between both HRTFs seamlessly to compare both. To ensure that an HRTF is not accidentally skipped, the system checks whether both HRTFs are activated at one point.

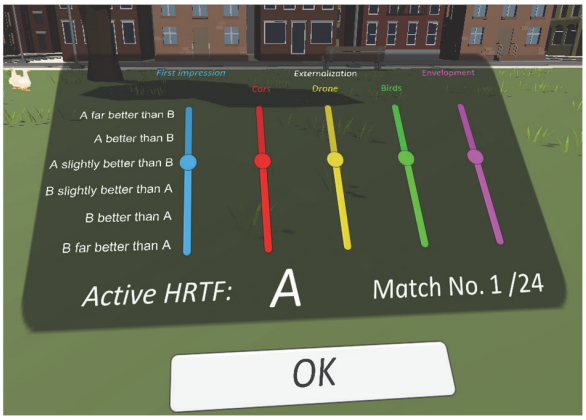


Figure 2. Screenshot of the display in the virtual environment of the tournament task.

After all matches have been played, the evaluation begins, in which the winners of the games are determined. Depending on the judgment, the correspondent HRTF gets a point from one (slightly better) to three (far better). While the "first impression" criterion is weighted with a factor of 1.5, the other criteria are weighted with 1. By comparing the sum of the points, the winner of the match is determined. In case both HRTFs have the same score, the *first impression* criteria decides the winner. The scoreboard is sorted by the number of winning matches. After that, the score for the criteria *first impression*, *externalization*, and *envelopment* is taken into account.

In the next round, the matches are paired based on the scoreboard to determine the next matches. The first place is matched against second place, the third place against the fourth place and so on. After six rounds with a total of 24 matches, the HRTF ranked first is considered the winner of the tournament and also the HRTF best suited for the participant.

2.4 Localization Task

After the tournament task, the winning HRTF, i.e., the HRTF ranked in first place, and the HRTF ranked last are used in a localization task. The personal simulated HRTF is always included in the test set, so a total of three HRTFs are tested in this task. If the personal HRTF ranks first or last, the HRTF that is next in rank to the personal HRTF is included.

The virtual environment is switched to an anechoic chamber resembling a real perception lab where loudspeaker models are arranged in a sphere around the user. Three hundred speakers are arranged in five planes and increments of 6 degrees horizontally and 14 degrees vertically. In addition to the virtual display, the representation of a dial in the loudspeaker array and a blue stripe on the front wall of the chamber facilitates the participant's orientation in the room and faster determination of the frontal direction (see Fig. 3). Before the test, the whole loudspeaker array adjusts its height to the participants head position, including an offset of 4 cm between the eye and the nose.

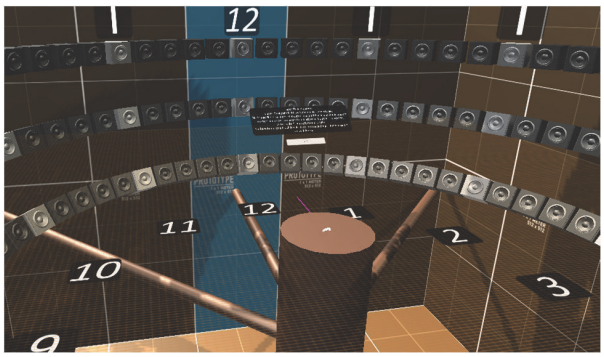


Figure 3. Screenshot of the virtual environment in the localization task.

For each HRTF, the participant is asked to determine 30 directions (see Fig. 4). The directions are given and their order is random. Also, no direction occurs more than once to make sure every direction is evaluated. The directions on the horizontal plane are identical for each layer to ensure a better comparison during evaluation. In addition, gaps have been added (front right and back left) to create a non-symmetrical distribution of the stimuli.

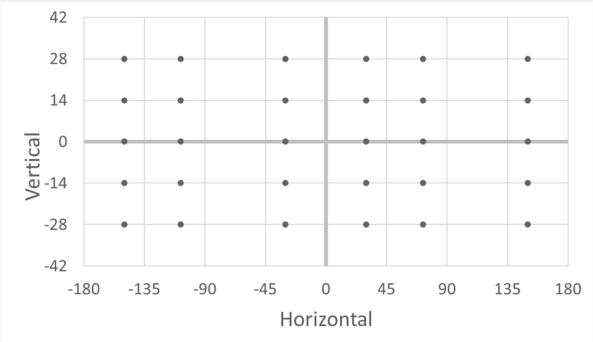


Figure 4. Distribution of the stimuli in the localization task (in degrees).

2.4.1 Test Procedure

The participant is asked to stay in the center of the loudspeaker array in the (virtual) anechoic chamber. During the trial, the sound will be muted in case they move away from the center and the distance from the center exceeds 50 cm. This is also visually implemented, as the floor is a transparent net with a circular platform in the center (see Fig. 3).

In the localization task, the participant has to judge from which loudspeaker the stimuli are originating from. A burst comprising pink noise pulses of 280 ms in length interleaved with silences of the same length is used as stimulus. To make their judgment, the participant marks the corresponding speaker with the VR controller. To avoid the user turning around and looking at the speaker to determine the origin of the sound, the stimulus is muted as soon as the difference between the participant's line of sight and the frontal direction exceeds 15°.

2.5 Ethical Approval

The study was conducted in accordance with the Declaration of Helsinki, with ethical approval obtained from Osnabrück University Ethics Committee (approval 4/71043.5). Anonymity of participants and confidentiality of their data were ensured. Participants were informed about the objectives and the procedure of the study as well as about their right to withdraw from the study at any time without adducing reasons or experiencing any negative consequences. All participants provided informed consent before participation in the study.

3. RESULTS

3.1 Tournament Task

As can be seen in Table 2, HRTF 0, 1, and 2 are overrepresented in the first two ranks, with HRTF 2 taking first place in four cases. The individual HRTFs were ranked in first place in two cases; the remaining HRTFs were ranked in the latter places, being second to last place for only one participant. While the positions in the midfield vary, i.e. different HRTFs can be found there depending on the participant, HRTF 3 (KEMAR HRTF), the only artificial head in the study, ranks last place in almost all cases. There is one exception where the HRTF of the KEMAR head is first and the personal HRTF is second to last.

Participant	HRTF							
	0	1	2	3	4	5	6	7
1	3	6	2	8	5	1	7	4
2	1	7	3	8	2	5	4	6
3	2	5	1	8	3	4	6	7
4	1	2	4	7	6	5	4	3
5	3	2	1	8	4	6	7	5
6	4	5	8	1	3	7	2	6
7	5	2	4	8	7	1	3	6
8	2	5	1	8	4	6	3	7
9	7	1	2	8	5	6	4	3
10	2	3	1	8	5	6	7	4

Table 2. Ranking of each HRTF in the tournament task (1 = first place, 8 = last place)

3.2 Localization Task

For each stimulus, the horizontal and vertical deviation between the stimulus direction and the participant's estimate of where the stimulus is coming from is calculated. Since the angular deviation is given only in positive values, the maximum deviation in the horizontal plane is 180°, and 56° in the vertical plane. As each participant had to determine the position of stimuli from combinations of 5 vertical and 6 horizontal positions, there were a total of 30 judgments.

Participant	HRTF with max. deviation in degree and [HRTF #]	HRTF with min. deviation in degree and [HRTF #]
1	52,8 [3]	36,8 [5]
2	47,2 [0]	29,4 [3]
3	39,6 [5]	31,6 [3]
4	48,6 [3]	26,4 [5]
5	56,8 [5]	43,0 [3]
6	46,2 [3]	35,8 [2]
7	24,8 [3]	17,6 [5]
8	34,0 [3]	30,4 [5]
9	55,0 [1]	49,8 [5]
10	48,8 [3]	30,0 [5]

Table 3. Average horizontal deviation for the worst HRTF (maximum deviation) and the best HRTF (minimum deviation) for the localization task. The individual numerically simulated HRTF is highlighted in grey. The specific HRTF is given in square brackets.

In a first evaluation, only the horizontal plane without elevation was considered, i.e., the mean of the horizontal deviations of all 30 judgments of each person was determined. The average deviation for the HRTF that had performed best in the localization test in each case is 33.1° for all 10 participants, and the average deviation for the HRTF that had performed worst in the localization test in each case is 45.4°. As in [20], deviations are consistently lowest around 90° and 270° and increase toward 180° and

360°. This also partly explains the relatively high deviation values on average. Moreover, a possible front-to-back confusion [34,35] is not taken into account in the calculation. It is noticeable that the winning HRTF from the tournament task was in no case the HRTF with which the best localization result was obtained, but the own (numerically simulated) HRTF is in 6 out of 10 cases the HRTF with which the best localization results are achieved (see Tab. 3). Although a chi-square test showed no significant results ($\chi^2(2, 10) = 3.800$, $p = .15$), probably mainly due to the small number of participants, the simulated HRTFs seem to have a positive effect in the localization task, in contrast to the tournament task.

In a second evaluation, the elevation was taken into account. For this purpose, the horizontal and vertical deviations were simply added for each stimulus. The average deviation for the HRTF that had performed best in the localization test in each case is 49.4°, and the average deviation for the HRTF that had performed worst in the localization test in each case is 60.4°. The results are almost identical as if only the horizontal deviation is considered. For only one participant (#10), the best performing HRTF in the localization test differed in this second evaluation, for whom their own (numerically simulated) HRTF was overperformed by the winning HRTF determined in the tournament task.

4. CONCLUSIONS

The results for the tournament task show that the numerically simulated individual HRTFs had little influence on the outcome, i.e., for the parameters *first impression*, *envelopment*, and *externalization* no advantage could be found for the participants' own HRTFs. Besides the limitation of explanatory power due to the relatively small number of participants in this explorative study, relevant issues could be (a) a possible bias due to the random selection of HRTFs from different databases, (b) the specific tournament mode (swiss style tournament), (c) the rating categories, (d) the visual/acoustic experimental environment within the VE or (e) problems due to the use of BCH. Since it has been shown in [18] and [20] that there is a significant correlation between tournament score and localization performance in a similar procedure with respect to (a), (b), and (c), the specific design of the acoustic environment seems to be important. For example, compared to [18] and [20], participants were free to move around the VE (including motion tracking) and the acoustic components of the soundscape were altered (motion path of car sounds, bird sounds present/absent, etc.). Combined with the use of BCH and the resulting poorer transmission of the higher frequencies, this may have led to the contradictory results. In a follow-up study, it would be beneficial to control these parameters in more detail, or to include them as independent variables. It could be particularly promising to use newer more advanced versions of BCH. The successor models Trekz Open Run and the recently released Open Run Pro have a significantly better transmission function, since higher frequency ranges are not only transmitted via bone conduction, but also partly hybrid as airborne sound. This presumably improves the added value of the individual HRTF in the scenario described here, while at the same time retaining the advantage of non-covered ears, which is particularly relevant for AR applications. In addition, it might be useful to extend the tournament task beyond the evaluation of a soundscape to other stimuli (speech, music, etc.).

However, the results of the localization task show that the individual numerically simulated HRTFs achieved the best results most of the time in the localization test. Although the positive influence of individual HRTFs on various perceptual capacities, in particular sound source localization tasks, has been frequently studied [1,10,11, 36,37], it is noteworthy that this also seems to hold for numerically simulated HRTFs created with a methodological approach oriented towards everyday usability. The overall relatively large angular deviations can probably also be attributed, at least in part, to the weaknesses of the BCH used and can most likely be improved by using newer models with more advanced procedures (see previous section). Another important point that should be addressed in future studies is the optimization of the scanning procedure as well as the post processing. For the intended suitability for everyday use, the scanning process must become more error-tolerant on the one hand, and it would be desirable to further automate post-processing on the other.

As an outlook, numerical HRTF simulation seems to be a promising approach for use in (acoustic) VE. Since the scanning process is always error-prone, several models of an individual could be simulated that differ in certain parameters. The best variant could then be determined in a perception-based tournament task. First, however, the tournament procedure needs to be improved because, although the simulated HRTFs provided a significant advantage in the localization task in the current study, they had little effect on the outcome of the tournament task.

Acknowledgments

The work is funded by the Volkswagen Foundation (VolkswagenStiftung) Germany within the project “New Concepts for Music and Media Technology” (grant no. 96 881).

5. REFERENCES

[1] Jenny, C. and Reuter, C., “Usability of individualized head-related transfer functions in virtual reality: Empirical study with perceptual attributes in sagittal plane sound localization,” JMIR Serious Games, 2020, 8(3), e17576.

[2] Armstrong, C., Thresh, L., Murphy, D., and Kearney, G., “A perceptual evaluation of individual and non-individual HRTFs: A case study of the SADIE II database,” Applied Sciences, 2018, 8(11), pp. 2029.

[3] Jenny, C. and Reuter, C., "Can I trust my ears in VR? Literature review of head-related transfer functions and valuation methods with descriptive attributes in virtual reality," International Journal of Virtual Reality, 21(2), pp. 29-43.

[4] Berger, C. C., Gonzalez-Franco, M., Tajadura-Jiménez, A., Florencio, D. and Zhang, Z., "Generic HRTFs may be good enough in virtual reality. Improving source localization through cross-modal plasticity," Frontiers in neuroscience, 2018, 12, pp. 21.

[5] Guezenoc, C. and Seguier, R., "HRTF individualization: A survey," arXiv preprint arXiv:2003.06183.

[6] Li, S. and Peissig, J., "Measurement of head-related transfer functions: A review," Applied Sciences, 2020, 10(14), pp. 5014.

[7] Gumerov, N. A., Duraiswami, R., and Zotkin, D. N., "Fast multipole accelerated boundary elements for numerical computation of the head related transfer function," In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07, Honolulu, 2007, Vol. 1, pp. 165-168.

[8] Prepeliță, S., Geronazzo, M., Avanzini, F., and Savioja, L., "Influence of voxelization on finite difference time domain simulations of head-related transfer functions," The Journal of the Acoustical Society of America, 2016, 139(5), pp. 2489-2504.

[9] Huttunen, T., Seppälä, E. T., Kirkeby, O., Kärkkäinen, A., and Kärkkäinen, L., "Simulation of the transfer function for a head-and-torso model over the entire audible frequency range," Journal of Computational Acoustics, 2007, 15(04), pp. 429-448.

[10] Mokhtari, P., Nishimura, R., and Takemoto, H., "Toward HRTF personalization: an auditory-perceptual evaluation of simulated and measured HRTFs," In Proceedings of the International Community for Auditory Display, 2008.

[11] Middlebrooks, J. C., Macpherson, E. A., and Onsan, Z. A., "Psychophysical customization of directional transfer functions for virtual sound localization," The Journal of the Acoustical Society of America, 2000, 108(6), pp. 3088-3091.

[12] Zotkin, D. N., Duraiswami, R., and Davis, L. S., Customizable auditory displays. Georgia Institute of Technology, 2002.

[13] Li, L. and Huang, Q., "HRTF personalization modeling based on RBF neural network," In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, May 2013, pp. 3707-3710.

[14] Iida, K., Ishii, Y., and Nishioka, S., "Personalization of head-related transfer functions in the median plane based on the anthropometry of the listener's pinnae," The Journal of the Acoustical Society of America, 2014, 136(1), pp. 317– 333.

[15] Seeber, B. U. and Fastl, H., "Subjective selection of non-individual head-related transfer functions," in Proceedings of the International Conference on Auditory Display, Boston, 2003, pp. 259-262.

[16] D. N. Zotkin, R. Duraiswami and L. S. Davis, "Rendering localized spatial audio in a virtual auditory space," in IEEE Transactions on Multimedia, 2004, 6(2), pp. 553–564.

[17] Shukla, R., Stewart, R., and Sandler, M., "User HRTF Selection for 3D Auditory Mixed Reality," In Proceedings of the Sound and Music Computing Conference, 2021, pp. 84-91.

[18] Voong, T. M. and Oehler, M., "Tournament Formats as Method for Determining Best-fitting HRTF Profiles for Individuals wearing Bone Conduction Headphones," In Proceedings of the 23rd International Congress on Acoustics integrating 4th EAA Euroregio, 2019, pp. 4841-4847.

[19] Iwaya, Y. "Individualization of head-related transfer functions with tournament-style listening test: Listening with other's ears," Acoust. Sc. Tech., 2016, 27(6), pp. 340-343.

[20] Voong, T. M., Reuter, C., and Oehler, M., "Influence of individual HRTF preference on localization accuracy–a comparison between regular and bone conducting headphones," In Proceedings of the Audio Engineering Society Convention 148, Vienna, May 2020, Audio Engineering Society.

[21] Stanley, R. M., "Measurement and validation of bone-conduction adjustment functions in virtual 3D audio displays," Doctoral dissertation, 2009, Georgia Institute of Technology.

[22] Chang-Geun, K. Lee, and P. Spencer, "Effectiveness of Advanced Bone Conduction Earphones for People Who Enjoy Outdoor Activities," In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 2011, 55(1), pp. 1788–1792.

[23] Ziegelwanger, H., Kreuzer, W., and Majdak, P., "Mesh2HRTF: Open-source software package for the numerical calculation of head-related transfer functions," in Proceedings of the 22nd International Congress on Sound and Vibration, Florence, 2015, IT.

[24] Ziegelwanger, H., Majdak, P., and Kreuzer, W., "Numerical calculation of listener-specific head-related transfer functions and sound localization: Microphone model and mesh discretization," The Journal of the Acoustical Society of America, 2015, 138, pp. 208-222.

[25] VIVE Pro Eye Overview | VIVE United States. https://www.vive.com/us/product/vive-pro-eye/overview/

[26] Unity Real-Time Development Platform | 3D, 2D VR & AR Engine. https://unity.com//

[27] M. P. Jenny C. and C. Reuter, "SOFA Native Spatializer Plugin for Unity - Exchangeable HRTFs in Virtual Reality," in Proceedings of the 144th Convention of the Audio Engineering Society, Milan, 2018.

[28] Roginska, A., Wakefield, G. H., and Santoro, T. S., "User selected HRTFs: Reduced complexity and improved perception," in Undersea Human System Integration Symposium, Providence, RI, USA, 2010, pp. 1–14.

[29] Shukla, R., Stewart, R., Roginska, A., and Sandler, M., eds. "User Selection of Optiman HRTF Sets via Holistic Comparative Evaluation," International Conference on Audio for Virtual and Augmented Reality 2018, Redmond, WA, USA.

[30] Gardner, W. G., and Martin, K. D., "HRTF measurements of a KEMAR," Journal of the Acoustical Society of America, 1995, 97, pp. 3907-3908.

[31] Algazi, V. R., Duda, R. O., Thompson, D. M., and Avendano, C., "The cipic hrtf database," in Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575), 2001, pp. 99–102.

[32] Warusfel, O., "Listen HRTF Database," May 2003. http://recherche.ircam.fr/equipes/salles/listen/index.html.

[33] Miller, J. D., Godfroy-Cooper, M., and Wenzel, E. M., "Using Published HRTFS with Slab3D: Metric-Based Database Selection and Phenomena Observed," presented at the 20th International Conference on Auditory Display (ICAD), New York, 2014, pp. 22-25.

[34] Cho, S. J., Ovcharenko, A., and Chong, U., "Front-Back Confusion Resolution in 3D Sound Localization with HRTF Databases," in International Forum on Strategic Technology, 2006, pp. 239-243.

[35] Park, M., Choi, S., Kim, S., and Bae, K., "Improvement of front-back sound localization characteristics in headphone-based 3D sound generation," in The 7th International Conference on Advanced Communication Technology - ICACT 2005, pp. 273-276.

[36] Wenzel, E. M., Arruda, M., Kistler, D. J., and Wightman, F. L., "Localization using non individualized head-related transfer functions," in Journal of the Acoustical Society of America 1993, 94, pp.111-123.

[37] Oberem J., Richter J. G., Setzer D., Seibold J., Koch I., and Fels J., "Experiments on localization accuracy with non-individual and individual HRTFs comparing static and dynamic reproduction methods", bioRxiv, 2020.

A Machine Learning Approach for MIDI to Guitar Tablature Conversion

Maximos Kaliakatsos-Papakostas[1], Grigoris Bastas[1,2], Dimos Makris[3], Dorrien Herremans[3], Vassilis Katsouros[1], Petros Maragos[2]
[1]Institute for Language and Speech Processing, Athena R.C., Athens, Greece
[2] School of Electrical and Computer Engineering, NTUA, Athens, Greece
[3] Department of Computer Science and Design Pillar, SUTD, Singapore
{maximos,g.bastas,vsk }@athenarc.gr, maragos@cs.ntua.gr,
{dimosthenis_makris,dorien_herremans}@sutd.edu.sg

ABSTRACT

Guitar tablature transcription consists in deducing the string and the fret number on which each note should be played to reproduce the actual musical part. This assignment should lead to playable string-fret combinations throughout the entire track and, in general, preserve parsimonious motion between successive combinations. Throughout the history of guitar playing, specific chord fingerings have been developed across different musical styles that facilitate common idiomatic voicing combinations and motion between them. This paper presents a method for assigning guitar tablature notation to a given MIDI-based musical part (possibly consisting of multiple polyphonic tracks), i.e. no information about guitar-idiomatic expressional characteristics is involved (e.g. bending etc.) The current strategy is based on machine learning and requires a basic assumption about how much fingers can stretch on a fretboard; only standard 6-string guitar tuning is examined. The proposed method also examines the transcription of music pieces that was not meant to be played or could not possibly be played by a guitar (e.g. potentially a symphonic orchestra part), employing a rudimentary method for augmenting musical information and training/testing the system with artificial data. The results present interesting aspects about what the system can achieve when trained on the initial and augmented dataset, showing that the training with augmented data improves the performance even in simple, e.g. monophonic, cases. Results also indicate weaknesses and lead to useful conclusions about possible improvements.

1. INTRODUCTION

Tablature constitutes a very old form of music notational language which provides guidance towards an active and intimate interconnection with the instrument, in contrast to music scores which aim at capturing more abstract psychoacoustic characteristics of sound, such as pitch or intervals [1]. A common type of tablature these days is guitar tablature (hereinafter simply referred to as tablature) and represents sequences of string-fret combinations that correspond to specific notes. It contains information about the fretboard positions on which the notes are articulated, which is not contained in typical music scores. However, tablature notation has limited expressive power when it comes to rhythmic structure and dynamics. It is this combination of features, though, which renders tablatures valuable in many occasions, since, assuming some auditory familiarity with the represented piece, they are more easily readable by non-experts. This can be important for general teaching purposes, especially among beginner and self-taught guitar players. Additionally, style-specific requirements have been established that would guide a transcriber of a piece to specific fingering choices, e.g., preference for lower-pitches strings when playing power-chords in metal music. Thus, a system that can generate tablatures, given the notes that make up a musical work, can be of great significance for the guitarist community. The difficulty of this task mainly arises due to the guitar's design, since same pitch notes can be played on more than one position on the fretboard. These facts make machine learning approaches relevant: how can a system learn from context what the best tablature is for a given set of pitches?

Related research includes generating tablature notation directly from audio, midi, or sheet music. Some research has focused on transcribing tablature directly from audio by developing accurate string classification models. Barbancho et al. [2] proposed a method to simultaneously detect the pitch and string on which a note should best be played by relying on inharmonicity analysis that enabled accurate partial tracking. Other research teams have used estimations of the inharmonicity coefficient in order to train bayesian statistical models [3, 4]. The inharmonicity coefficient has also been used as one of the input features to classification models, such as Support Vector Machines (SVMs) [5,6]. The above strategies focus mainly on monophonic performances with the exception of [2] who incorporate chord recognition within certain limits.

Some research on tablature transcription has focused on deducing plausible sequences of string-fret combinations relying mostly on playability constraints. They dispose of string-specific audio features, which are difficult to acquire in polyphonic performances, and practically reduce the problem to two phases: (multi-)pitch detection followed by pitch to tablature conversion. Barbancho et. al [7] pro-

Copyright: © 2022 First author et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

posed a system for chord and fingering recognition using Hidden Markov Models (HMMs) that encode the probabilities to move from one configuration to the other. Burlet and Fujinaga [8] encoded string-fret transitions using weighted directed acyclic graphs (WDAGs) and the A^* algorithm was employed to determine the optimal sequences. In [9] and [10], where the audio was produced by MIDI, a graph-based representation was employed using WDAGs, and a dynamic programming (DP) algorithm was applied to find longest paths.

Several other research teams have worked on fingering analysis, playable guitar tablature generation, and fingering recognition given a series of note events in symbolic notation (music scores or MIDI) either by employing graph representations, Dynamic Programming (DP) strategies [11–15], or other optimization algorithms [16–21]. For example, Tuohy and Potter [16] proposed a simple genetic algorithm (GA) which incorporates limitations related to guitar playability in a fitness function. The same team has managed to combine Artificial Neural Networks (ANNs) with a GA [17] for the problem of musical score to tablature conversion. In [18] they also ascribe fingerings to the predicted string-fret positions.

ANNs have also been employed to tackle the problem of tablature estimation directly from audio signals. Multilayer Perceptrons [22] and Convolutional Neural Networks (CNNs) [23] have been used for chord recognition and the detection of hand positions corresponding to chords. More recently, Wiggins and Kim [24] applied CNNs for string-fret recognition in solo performances that which also included elements of polyphony. Deep learning methods were also recently applied as a method for guitar tablature composition (generation) [25], more specifically, using a Transformer.

2. METHOD

The proposed method [1] assumes normal-tuning guitars with six strings and 24 frets – 25 values per string are considered for representing the free string as well. The transcription is considered to involve only information about what midi pitches need to be transcribed, disregarding information about inter onset intervals between successive sets (or frames) of pitches to be transcribed; also midi velocity and duration are not considered. Therefore, the transcription task is described as a "binary process", where a binary representation of input midi pitches is transcribed into a binary representation of a tablature. The overall method is divided in two parts for reaching a single tablature decision: a) a deep neural network is employed for generating a "probabilistic" tablature that indicates which string-fret combinations are more probable for a given input and b) a search algorithm that uses a simple conceptualisation of guitar "playability" for selecting the best string-fret positions given the previous finger positions and taking into account the highest-probability tablature frame.

The aim of the overall method is to assign a playable tablature notation to a given set of notes (midi), whether or not the complete set of these notes is playable on a guitar fretboard, e.g., because there are more than six simultaneous notes or because the notes stretch on a wide octave range that cannot be covered by a guitar. During the application of the algorithm in the second part of the method, which involves examining all fretboard combinations that implement the input midi pitches, it is possible to iteratively discard midi pitches from being considered for transcription. This can either happen because the midi input frame might involve more than six pitches (which are impossible to be played simultaneously in a six-string guitar) or because of improper pitch layout (e.g., two pitches that correspond to the 1st and 2nd string fret of the lower E string). It is also assumed that guitar fingering on the tablature does not only depend on the current (midi) pitches to be played, but also from previous finger positions on the fretboard, or previous fretboard frames. Motion of the fingers on the fretboard between successive tablature frames, accounts for minimizing the travel distances across the fretboard and for preserving fingering shapes between successive frames as much as possible.

2.1 Probabilistic Fretboard Deep Neural Network

The deep neural network that was developed for solving the problem of assigning tablature to midi input manages to model fingering shapes and temporal dependencies between successive tablature frames. The architecture is shown in Figure 1. Fingering patterns are expected to be learned from data. A Convolutional Neural Network (CNN) architecture is choice for doing so, since convolutional filters are designed to capture spatial patterns, e.g., on a 2D fretboard. Specifically, the output of the network employs two layers of transposed convolutions (or deconvolutions) that result in the formation of a 6×26 fretboard with 6 strings and 25 frets plus the free string (there is a lambda layer dedicated to simply dropping the 25th fret information). The first layer of deconvolutions (top right box in Figure 1) generates a 3×12 structure that can be thought of as "blurry", or highly pixelated, low resolution version of a fretboard image. This layer learns to describe filter combinations in the next deconvolution layer, which actually forms the final version of the aforementioned 6×26 fretboard.

Input to these two deconvolutional layers is a "latent" space with 384 dimensions, which is formed by successive transformations of three feedforward layers (with scaled exponential linear activation) of the overall input to the network. The input of the network includes not only the midi pitches to be transcribed to tablature, but also the 4 previous frames of the tablature, as a crude approach to involving information about successive frame dependencies. The input is a concatenated array that incorporates this information, of total size 728: 128 midi pitches to be transcribed plus 600 (4 times 125) for the four previous tablature frames. All input is a binary vector, with ones representing the index of active midi pitches to be transcribed and active string-fret combinations in each input tablature frame.

During the training phase, the network learns to approx-

[1] https://github.com/maximoskp/midi2tab_Deconvolutions

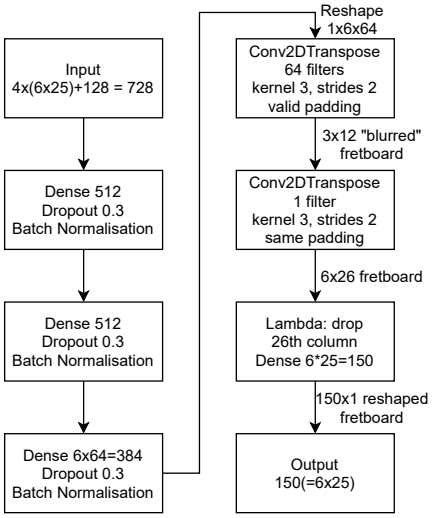


Figure 1. Proposed architecture.

imate the target tablature (in binary representation) with output activity of the deconvolutional layers. Even though no encoder-decoder parts are explicitly constructed, it might be helpful to think of the vertical alignment of boxes on the left part of Figure 1) as the encoder, which generates a latent representation of the input with 384 elements, and the right part as the decoder, which generates a "*probabilistic tablature*" based on the latent space generated by the input. The term "probabilistic tablature" is employed for denoting the fact that the network is able to generate an approximation of the tablature that is not binary; i.e., it incorporates real values that indicate (without forming a probability density function that sums to one) the probability of a string-fret pair being active.

2.2 Probabilistic Fretboard to Actual Fingering

This section describes the process of assigning actual fingerings (or fretboards – terms are used interchangeably) based on the "probabilistic fretboard" produced by the network. The algorithm for keeping all possible and playable fretboards from a set of midi pitches $\vec{m}$ is summarised as follows:

1. Bring all pitches of $\vec{m}$ within the fretboard range, by increasing or decreasing their pitch values by an octave.
2. Decide on the number of pitches from $\vec{m}$ to be preserved, which is either the number of pitches $\vec{m}$ or 6, if this number exceeds 6 (since only 6-string guitars are considered). This number is symbolized as N.
3. Keep all combinations of pitches of $\vec{m}$ that include N notes.
4. For each combination, generate all possible binary fretboards and keep the ones that are "*playable*". A binary fretboard is considered to be *playable* if: a) there is at most one pitch per string and b) all non-open string pitches fall within a window of 6 frets (considering that finger stretching of over 6 frets is not acceptable).
5. If there is no binary fretboard that satisfies the above mentioned criteria, reduce N by one and go back to step 3.

After all possible binary fretboards are obtained, denoted as $b_k \in \{0,1\}^{6\times 25}$, $k \in \{1,2,\ldots,K\}$, where K is the number of retrieved possible binary fretboards, the fretboard that corresponds to the higher sum of values in the probabilistic tablature provided by the network ($p \in \mathbb{R}^{6\times 25}$) is retained. This sum is obtained through the inner product of the probabilistic tablature and each examined binary tablature:

$$b = \underset{k}{\operatorname{argmax}}(p \cdot b_k), \tag{1}$$

where $\cdot$ denotes the inner product of b_k and p. Figure 2 depicts the decision-making process, where the top graph shows the probabilistic tablature generated by the network, the bottom shows the ground-truth tablature and the middle part shows the binary tablature, among all the possible ones for the pitches to be transcribed, that achieves maximum value of the inner product in eq. 1. The selected binary tablature (middle) is the same as the ground-truth tablature (bottom) in this case.

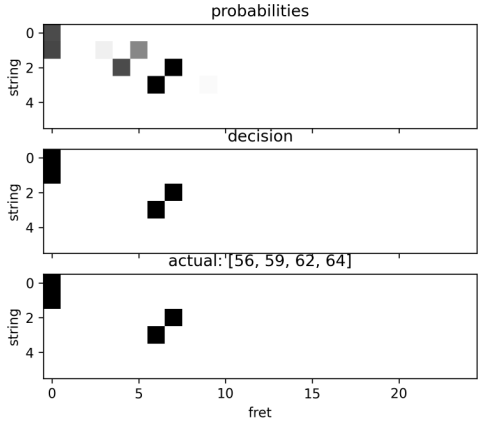


Figure 2. Probabilistic fretboard (bottom), final decision (middle) and comparison with ground truth (bottom).

3. DATASET AND DATA MANIPULATION

The DadaGP dataset [26] was used as dataset for the current paper. This dataset contains 26,181 pieces in GuitarPro (GP) format, i.e., in tablature notation for guitar and string instruments, and those pieces are divided in 739 genres. Even though an encoded form of the dataset in tokenized form is also available, the study at hand employs the original GP files for the necessary conversions presented in this section. Data in the system was represented in a binary format (described in the next paragraph) that generated multiple instances per piece, making the entirety of the dataset practically unusable because of its large size. Therefore, a 5% part of the entire dataset was employed, in which a random 5% of pieces from each folder in the DadaGP was included. Since the folder structure in DadaGP divides pieces per album, it is expected that this data selection method creates a set with a distribution among all available styles that is similar to the distribution in the original set. Additionally, the approach presented

herein discusses only transcription for 6-string guitars in standard tuning, excluding all piece that employ different tunings and number of strings. This study also excludes guitar-idiomatic and expressional information, e.g., string bending, hammer-on, pull-off etc. Even though this information is crucial for guitar transcription, it is left for future work.

From the pieces that are retained, only the guitar tracks are preserved. From each guitar track, discrete successive events are extracted, where each event includes pitch-related fretboard information for any change that occurs on the fretboard. Each event represents an instance of a fretboard layout at a given time, where potentially multiple notes are active, each note on a separate string (i.e., at most 6 notes can be played simultaneously). A fretboard layout, in turn, is represented as a 6-by-25 binary matrix, where rows represent strings and columns 24 (plus a free string) frets. In each binary matrix, occurrences of the digit 1 indicate which pitch is played on which fret (first columns of each string indicate a free string); each binary matrix is hereby referred to as a " tablature frame" and each piece includes so many frames as the number of tablature changes it includes. Each tablature frame is then transcribed to the respective binary midi frame; primary aim of the presented method is to develop a method for translating midi frames to tablature frames.

Tablature frames are reshaped to row-shaped arrays with 150 (6×25) elements and each piece is represented as a stack of successive frames on top of each other; similarly with midi frames. These two stacks of frames are show in Figure 3. Each piece is also padded with a number of all-zero frames, depending on the assumed tablature "history" considered in the model, i.e., how many previous frames should be considered for identifying fingering patterns that are more probable at any given instance. Tablature history is assumed an important factor when deciding what the next fingering layout should be. Therefore, tablature history is included in the design of the examined model. To this end, and according to Figure 3, considering a history of four frames, the input to the system is the tablature included in the past four frames (600 binary numbers) along with the current midi to be transcribed in the current frame (128 binary numbers, totalling an input of 728 numbers). The system learns to generate output that predicts the current tablature frame. Summarising, the model learns to transform an input of 728 binary digits to 150 binary digits. The 5% of the DadaGP dataset that is retained totals a number of 955971 training, 239402 validation and 299423 test time frames.

Another aim is to study whether the presented method is flexible enough to transcribe midi input that is not playable by a single guitar. Since there is no dataset incorporating such information, a naive music-drive data alteration method is followed which "augments" midi information. This augmentation (in terms of musical information) method assumes that any tablature frame might have been a result of transcribing richer (in terms of number of pitches) midi content. To this end, guitar frame transformation from tablature to midi involves the insertion of additional pitches. Specifically, for each pitch in the tablature, there is a 50% chance that a new midi pitch will be inserted an octave up, down, a fifth, a fourth, a major/minor third or a major/minor sixth (selecting uniformly between these notes). The midi input will therefore include more pitches than the ones that appear in the target tablature, forcing the network to learn to ignore midi pitches that do not fit into the context of guitar transcription.

The intention is to make the system selective towards tablature interpretations of midi input that are more convenient for guitar playability, allowing the possibility to exclude pitches that do not offer much in terms of musical information and, at the same time, would possibly result to fingerings that are less frequent in the dataset. This data augmentation method has severe weaknesses, especially in cases where the tablature frame involves few notes. For instance, if the tablature frame includes one pitch, then augmenting the respective midi frame with, e.g., a major third up would create a pair of notes that would be easily playable with a guitar and, in fact, there are still multiple instances of this pair of notes in the original, not augmented dataset. This potentially leads to multiple conflicting tablature outputs for the same midi input, where in one case, the non-augmented midi input would correspond to the exact fingering in the target tablature, while in another case, the augmented midi output would (possibly) be the same as in the first case, but would correspond to a different tablature. This method could be further refined, e.g., to allow augmentation only in cases where a number of simultaneous pitches above a predefined threshold is involved (e.g., augmentation could occur to tablature frames that include more than four notes). The simplest approach, however, that possibly adds pitches in any frame, presents results that are interesting, so this augmentation method is used in the reported results in this paper.

4. RESULTS

The method is examined in terms of learning adaptivity for the neural network and in terms of actual transcription characteristics for the entire system. The learning errors of the neural network provide an indication about how well the "probabilistic tablature" output generated by the network approaches the target tablatures as the epochs increase in terms of the mean square error, but these errors do not provide an intuitive assessment of how clearly this output can be translated into the binary target tablatures at each epoch. Ideally, a complete binary output of the system should be employed to evaluate the accuracy of the system, however, it is prohibitively slow to apply the analytical step for deciding the best binary tablature for any given probabilistic tablature (input data point), in each epoch, in the training and the evaluation steps.

To this end, a fast to compute and intuitive measurement of how well the probabilistic tablature output is aligned with the target binary output is the cosine similarity, given by

$$\cos(\theta) = \frac{\vec{p} \cdot \vec{b}}{||\vec{p}||\,||\vec{b}||}, \tag{2}$$

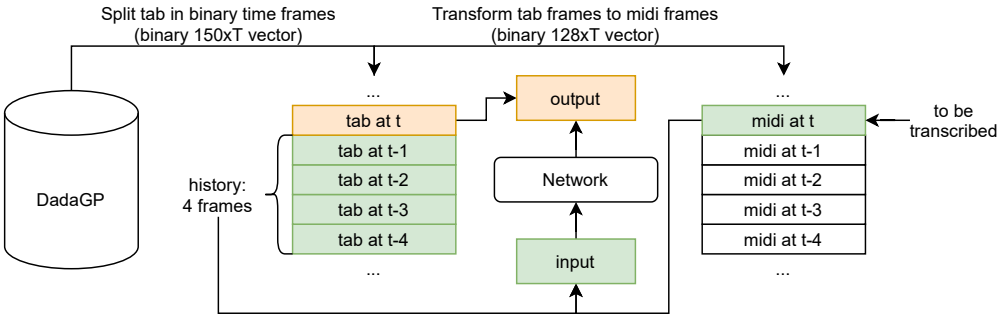


Figure 3. Data preparation overview.

where $\vec{p}$ is the probabilistic tablature generated by the network and $\vec{b}$ is the target binary tablature. This measure computes the dot product over the product of the norms of each vector, showing, intuitively, how well the predictions are aligned (or correlated) with the (binary) target, regardless of the magnitude in these predictions. In the analytical step of the tablature generation process, the $\vec{p}$ produced by the network is compared with all possible binary tablatures $\vec{b_k}$ through the dot product (eq. 1). The difference between eq. 1 and eq. 2 is that in the latter, the inner product is normalized with the norm of both involved vectors, while in the former this is not necessary, since the maximum value is assessed and all binary tablature vectors have the same norm (all include the same number of ones).

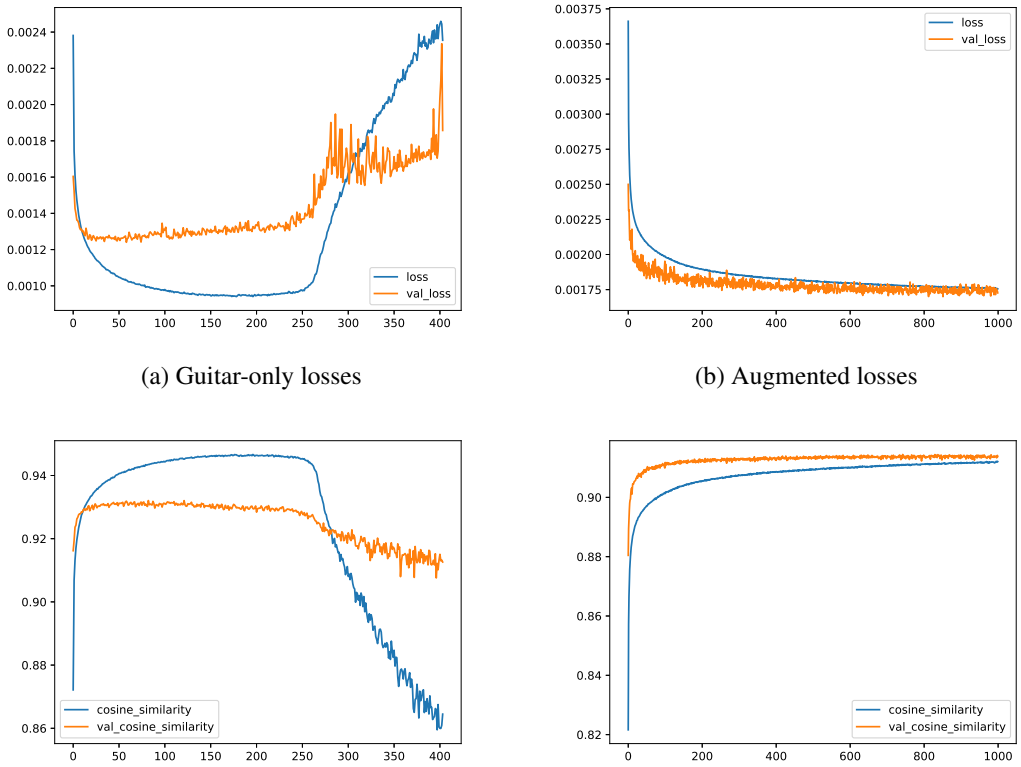
Figure 4 shows the losses and the cosine-based measured accuracy in the training and validation sets for the guitar-only and augmented datasets. One thing that can be immediately noticed is the difference in both loss and accuracy during training for the two datasets. The network, with the guitar-only dataset seems to be reaching the highest learning capacity too early. Training, in both dataset cases, is stopped at the epoch with the lowest validation set loss (orange lines in (a) and (b) graphs of Figure 4) to avoid overfitting. Table 1 shows that this point is reached in epoch 59 for the guitar-only dataset and 852 for the augmented dataset. It should be noted that the plotted curves in the guitar-only case are shown up to around half the epochs in comparison to the augmented dataset case, where all 1000 epochs are shown.

	guitar data	augmented data
epoch	59	852
train. loss	1.029×10^{-3}	1.767×10^{-3}
val. loss	1.259×10^{-3}	1.775×10^{-3}
train. accuracy	0.941	0.911
val. accuracy	0.931	0.913

Table 1. Epoch, loss and validation values when optimal validation accuracy was achieved (see Figure 4).

Even though cosine similarity is a useful indication about how accurate the estimations performed by the network are, the actual tablature produced after following the analytic step is not consider. To address this issue, the test set is employed (different from the training and evaluation sets) for examining the entire system, from input to a single binary output decision. The effectiveness of the entire system is cross-examined with the network trained and running on the guitar-only and the augmented data. A possible candidate for assessing the effectiveness of the system in terms of binary output is through precision, recall and f-measure. This method, however, would not be very informative towards interpreting the system decisions in the problem at hand. One reason is that precision and recall, and thus the f-measure, are tied with a strict correspondence. Since the number of ones in the system-generated binary tablature and the ground-truth binary output are constant, for any given example the false positives will always be the same as the false negatives, i.e., any 1 value that the system fails to recognise (false negative) will always be misplaced, producing a 1 value at a wrong position (false positive). Another reason is that there is no actual ground-truth in the augmented dataset, since it was constructed by adding pitches never existed in the tablature of the dataset. Therefore, in the cases where pitches were added, it would be normal to have a maximum precision value below 1.

More useful information about how accurately the system performs are extracted through measuring the percentage of cases the output exactly matched the ground-truth data, i.e., the binary output is exactly the same as the target binary tablature, in how many cases there was a partial match, i.e., how many times at least one pitch matched a ground-truth string and fret, and in how many cases there was no match, i.e., no pitch was assigned to the ground-truth string and fret. It should be noted that when the augmented dataset is tested, in the cases where pitch augmentation has occurred, it is most likely that no exact match can be found. If the tablature under examination has been augmented, the system is asked to compute a tablature that is indeed different from the initial ground-truth tablature, since additional pitches have been added and requested to be included in the output tablature. This means that the correct system response will be to produce a tablature that accommodates all requested pitches, given that they can be accommodated within the "**playable**" margins (as defined in step 4 in Section 2.2). Even in the case of examining an augmented binary fretboard, it could be actually possible to end up with a set of pitches for transcription that is the same with the set of pitches in the initial, non-augmented, fretboard, if the pitch reduction process in step 3 happens



(a) Guitar-only losses

(b) Augmented losses

(c) Guitar-only cosine accuracy

(d) Augmented cosine accuracy

Figure 4. Losses and accuracy for all training epochs in the guitar-only and augmented transcription problems (see Table 1 for optimal conditions).

to lead to the set of pitches included in the initial fretboard; this, however, would be rare.

Table 2 shows the cross-comparison between the two versions of the system, i.e., trained with guitar-only and augmented data. Both versions are examined by running on the both the guitar-only and the augmented test sets, in 5000 random tablature frame assignment tasks. Information in this table is given for all number of pitches on the fretboard, to show that in the guitar-only test dataset there is a clear dominance (70%) of single-pitch instances to be transcribed. This dominance fades out with the augmentation process, since 50% of those single-pitch instances obtain a second, added pitch. It is clear that the system trained with the augmented dataset performs better not only when tested on the augmented dataset (62% vs 55% partial plus absolute matches), but also on guitar-only dataset (55% vs 44% partial plus absolute matches). The fact that both versions perform better in the augmented rather than the guitar-only test sets, can be mainly attributed to the fact that the guitar-only dataset includes much more single-note instance, where both versions exhibit exceptionally bad performance (especially the guitar-only trained version has 45% failures and 25% successes).

Another problem that has been observed concerns the analytical step in the augmented dataset tests. This step has been designed to be "greedy", i.e., it starts by trying to fit as much of the requested pitches on the fretboard as possible. For example, if more than size pitches are requested, this step will first generate and examine all combinations that comprise playable solution of six pitches, regardless of how probable (as a fingering in the dataset) they are. If such a playable combination is identified, even with a near-zero probability (according to eq. 1), this will be returned as a solution. In some cases, this strategy is not optimal; such an example is illustrated in Figure 5. The caption of the bottom graph ("actual" pitches) shows that eight pitches are requested and the system is able to accommodate six of them in a playable combination on the fretboard (middle graph). The probabilities that correspond to those fretboard pitches, however, are close to zero, as indicated by the top graph, which shows the network probabilistic tablature output. In fact, the network has been trained so efficiently, that regardless of the fact that eight pitches are requested in the input, the probabilistic tablature output follows a clear four-string major or minor pattern that is very close to the pre-augmentation ground-truth tablature (bottom graph). Therefore, in this case, it would possibly be better for the analytical step to continue examining fingerings with less than six pitches, even though such a fingering was found, since fingerings with less pitches could, in fact, lead to tablatures with higher probabilities.

num. p.:	1	2	3	4	5	6	sum	1	2	3	4	5	6	sum
Guitar-only training - Guitar-only test								Augmented training - Guitar-only test						
no match	0.45	0.05	0.03	0.01	0.01	0.0	0.55	0.36	0.04	0.02	0.01	0.01	0.0	0.44
partial	0.0	0.03	0.05	0.01	0.0	0.0	0.09	0.0	0.03	0.03	0.01	0.0	0.0	0.07
match	0.25	0.05	0.03	0.01	0.0	0.01	0.35	0.34	0.06	0.05	0.01	0.01	0.01	0.48
sum	0.7	0.13	0.11	0.03	0.02	0.01		0.7	0.13	0.11	0.03	0.02	0.01	
Guitar-only training - Augmented test								Augmented training - Augmented test						
no match	0.16	0.16	0.02	0.03	0.03	0.05	0.45	0.12	0.15	0.02	0.03	0.03	0.05	0.40
partial	0.0	0.13	0.06	0.07	0.06	0.07	0.39	0.0	0.15	0.06	0.07	0.06	0.08	0.42
match	0.13	0.01	0.0	0.0	0.0	0.0	0.14	0.17	0.01	0.0	0.0	0.0	0.0	0.18
sum	0.29	0.31	0.09	0.1	0.09	0.12		0.29	0.31	0.09	0.1	0.09	0.12	

Table 2. Cross-comparison results as a ratios over the total number of examined data instances when model was trained and tested with guitar-only and augmented data, when the transcription task incorporated 1 to 6 number of pitches.

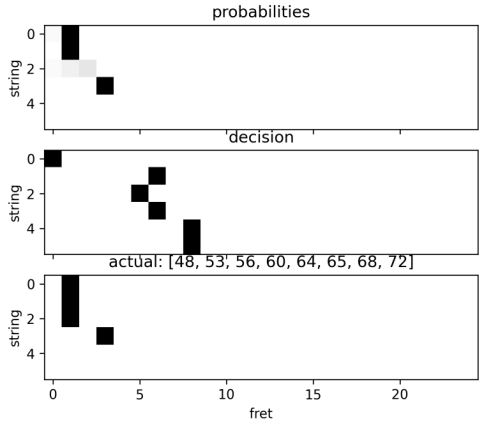


Figure 5. Decision (middle) is problematic since it tries to accommodate as many pitches as possible, regardless of how much they correlate with the probabilistic fretboard (top), which is well-aligned with the actual ground truth (bottom).

5. CONCLUSIONS

This paper presents a machine learning method for transcribing midi pitch information to guitar tablature. The method is based on neural networks that perform deconvolutions at the final stage; a part of the DadaGP was employed for learning and examination purposes. Conversion of musical parts that cannot possibly be played by a guitar was also examined, through a method that augments data artificially. Results indicate that training on artificial data, even though problematic in specific cases, generally led to better performance, even in single-pitch cases, where augmentation could not have happened. Based on the results, the analytic part of the method should be improved, so that fingerings that do not necessarily incorporate the maximum number of pitches can be accepted. Additionally, among the first improvements that should be attempted to the core mechanism of the method, is to incorporate information not only about the previous tablatures (past information), but also about the next possible tablatures (future information). This could be performed either by employing bidirectional LSTM or transformer components in the architecture.

Another improvement to the method would be to incorporate the examination of octave alteration of pitches that are not playable, instead of simply rejecting them. This would allow the accommodation of octave-equivalent information for as many pitches as possible in the transcription. However, such a process would require further "stream" information of octave-transposed pitches, something that would require identification of the melodic stream [27] where these pitches belong, along with examination of octave transposition to all the pitches within this stream. This is a complicated issue with implications in both the probabilistic and combinatorial parts of the proposed method, but the results would be much more valuable for transcribing pieces that have not been composed for and/or are not playble with guitar. Application-level improvements could include the possibility for selecting other guitar tunings, or generating exercises on given pitches with different fingering characteristics.

Acknowledgments

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (NLP-Theatre, project code: T1EDK-00508). This work is supported by Singapore Ministry of Education Grant no. MOE2018-T2-2-161.

6. REFERENCES

[1] J. Kojs, "Notating action-based music," *Leonardo Music Journal*, pp. 65–72, 2011.

[2] I. Barbancho, L. J. Tardon, S. Sammartino, and A. M. Barbancho, "Inharmonicity-based method for the automatic generation of guitar tablature," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1857–1868, 2012.

[3] J. Michelson, R. Stern, and T. Sullivan, "Automatic guitar tablature transcription from audio using inharmonicity regression and bayesian classification," in *Au-*

dio Engineering Society Convention 145. Audio Engineering Society, 2018.

[4] J. M. Hjerrild and M. G. Christensen, "Estimation of guitar string, fret and plucking position using parametric pitch estimation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 151–155.

[5] J. Abeßer, "Automatic string detection for bass guitar and electric guitar," in *International Symposium on Computer Music Modeling and Retrieval*. Springer, 2012, pp. 333–352.

[6] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score-and instrument-related parameters." in *DAFx*, 2014, pp. 219–226.

[7] A. M. Barbancho, A. Klapuri, L. J. Tardón, and I. Barbancho, "Automatic transcription of guitar chords and fingering from audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 915–921, 2011.

[8] G. Burlet and I. Fujinaga, "Robotaba guitar tablature transcription framework." in *ISMIR*, 2013, pp. 517–522.

[9] K. Yazawa, D. Sakaue, K. Nagira, K. Itoyama, and H. G. Okuno, "Audio-based guitar tablature transcription using multipitch analysis and playability constraints," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 196–200.

[10] K. Yazawa, K. Itoyama, and H. G. Okuno, "Automatic transcription of guitar tablature from audio signals in accordance with player's proficiency," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3122–3126.

[11] S. I. Sayegh, "Fingering for string instruments with the optimum path paradigm," *Computer Music Journal*, vol. 13, no. 3, pp. 76–84, 1989.

[12] A. Radisavljevic and P. F. Driessen, "Path difference learning for guitar fingering problem," in *ICMC*, vol. 28, 2004.

[13] D. Radicioni and V. Lombardo, "Guitar fingering for music performance," in *ICMC*, 2005.

[14] M. Miura, I. Hirota, N. Hama, and M. Yanagida, "Constructing a system for finger-position determination and tablature generation for playing melodies on guitars," *Systems and Computers in Japan*, vol. 35, no. 6, pp. 10–19, 2004.

[15] G. Hori, H. Kameoka, and S. Sagayama, "Input-output hmm applied to automatic arrangement for guitars," *Information and Media Technologies*, vol. 8, no. 2, pp. 477–484, 2013.

[16] D. R. Tuohy and W. D. Potter, "A genetic algorithm for the automatic generation of playable guitar tablature," in *ICMC*, 2005, pp. 499–502.

[17] D. R. Tuohy, W. D. Potter, and A. I. Center, "An evolved neural network/hc hybrid for tablature creation in ga-based guitar arranging," in *ICMC*. Citeseer, 2006.

[18] D. R. Tuohy and W. D. Potter, "Generating guitar tablature with lhf notation via dga and ann," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2006, pp. 244–253.

[19] D. R. Tuohy and W. Potter, "Guitar tablature creation with neural networks and distributed genetic search," in *Proc. of the 19th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA-AIE06, Annecy, France*, 2006.

[20] J. V. Ramos, A. S. Ramos, C. N. Silla, and D. S. Sanches, "Comparative study of genetic algorithm and ant colony optimization algorithm performances for the task of guitar tablature transcription," in *2015 Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2015, pp. 228–233.

[21] ——, "An evaluation of different evolutionary approaches applied in the process of automatic transcription of music scores into tablatures," in *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2016, pp. 663–669.

[22] T. Gagnon, S. Larouche, and R. Lefebvre, "A neural network approach for preclassification in musical chords recognition," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2. IEEE, 2003, pp. 2106–2109.

[23] E. J. Humphrey and J. P. Bello, "From music audio to chord tablature: Teaching deep convolutional networks toplay guitar," in *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2014, pp. 6974–6978.

[24] A. Wiggins and Y. Kim, "Guitar tablature estimation with a convolutional neural network." in *ISMIR*, 2019, pp. 284–291.

[25] Y.-H. Chen, Y.-H. Huang, W.-Y. Hsiao, and Y.-H. Yang, "Automatic composition of guitar tabs by transformers and groove modeling."

[26] P. Sarmento, A. Kumar, C. Carr, Z. Zukowski, M. Barthet, and Y.-H. Yang, "Dadagp: A dataset of tokenized guitarpro songs for sequence models," *arXiv preprint arXiv:2107.14653*, 2021.

[27] E. Cambouropoulos, "Voice and stream: Perceptual and computational modeling of voice separation," *Music Perception*, vol. 26, no. 1, pp. 75–94, 2008.

DEEP CONVOLUTIONAL OSCILLATOR: SYNTHESIZING WAVEFORMS FROM TIMBRAL DESCRIPTORS

Gordan Kreković
Visage Technologies
gordan.krekovic@visagetechnologies.com

ABSTRACT

This paper presents a novel deep learning model for synthesizing single-cycle waveforms from timbral attributes. The motivation was to investigate a viable alternative to traditional wavetable oscillators with intuitive control. Based on a thorough literature review and practical considerations, we selected three attributes appropriate for describing timbral characteristics of steady and harmonic tones: *bright*, *warm*, and *rich*. A deep learning network was designed to map magnitudes of these attributes to single-cycle waveforms. The architecture was based on stacking of upsampling and convolutional layers to model temporal dependencies within the waveform. The network was trained on a large number of waveforms extracted from NSynth dataset. Audio features closely related to the selected attributes were used as inputs, while the custom loss function was employed to minimize the difference in normalized power spectra between outputs and training waveforms. Four models with different hyperparameters were trained and the best one was selected using the validation dataset. Further experiments with the selected model showed that synthesized waveforms generally match the input attributes well, as the mean absolute errors for normalized attributes were 0.07, 0.05, and 0.18 for *bright*, *warm*, and *rich* respectively on the testing dataset.

1. INTRODUCTION

As it has happened in many fields during the last decade, deep learning has radically influenced automated generation of sound and music. Novel models such as MidiNet [1], MuseGan [2], MusicAutobot [3], and Music Transformer [4] demonstrated an undoubted dominance of deep learning over traditional approaches in generating polyphonic music sequences, while models such as SampleRNN [5], WaveNet [6], MelNet[7] and GANSynth [8] opened an exciting world of possibilities related to autonomous synthesis of raw audio. Recent advances in the field [9] and efforts in establishing relevant datasets [10] suggest that generative deep learning might successfully substitute or complement traditional sound synthesis methods in the upcoming years.

These promising trends motivated the central idea of this research – to build a waveform generator with interpretable controllable parameters using deep learning. The reasons behind choosing wavetable synthesis and interpretable parameters are multiple.

1.1 Wavetable Synthesis

Although idiosyncratic to the early years of digital sound synthesis, wavetable synthesis has been recently regaining some of its past glory. This synthesis method based on the principle of periodic reproduction of single-cycle waveforms [11, 12] is an appealing addition to other types of oscillators in comprehensive architectures of modern synthesizers, due to its capability of producing a wide range of steady, harmonic tones whose timbers can change over time. The evolving timbral character can be achieved by interpolating between multiple waveforms from a wavetable and by applying different modulations to the synthesized audio signal.

Instead of reading predefined waveforms from memory, a generative oscillator based on deep learning could act as an inexhaustible, yet controllable source of waveforms which can simply substitute a traditional oscillator without affecting other components of the sound synthesizer. The synthesized waveforms can still pass through the same amplifiers and filters controlled by the same low-frequency oscillators and envelope generators. A recent study [13] introduced the concept of differentiable wavetable synthesis in which the entire synthesis process has been turned to a differentiable digital signal processing pipeline.

The notion of substituting a standard wavetable oscillator with a generative model also motivated our previous research focused on building an autoencoder for waveform generation [14]. The autoencoder was trained to reconstruct single-cycle waveforms, while its latent variables served for traversing through the waveform space, enabling interesting transitions within the synthesized sound. The downside of that approach is that latent variables lack interpretability and intuitiveness, as they do not have an easily understandable relationship with the acoustical consequence they produce. For that reason, this research takes a different direction and examines an alternative deep learning architecture, together with different training and evaluation processes aimed at achieving a generative oscillator with more intuitive parameters.

Copyright: © 2022 Gordan Kreković. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any médium, provided the original author and source are credited.

1.2 Interpretability and Intuitiveness

Even though the lack of intuitiveness especially concerns autoencoder's latent variables, parameters of traditional synthesis methods may also pose a challenge to efficient and inspired sound design. Synthesis parameters do not necessarily bear acoustical meaning and sometimes depend on each other, so they require technical knowledge and a certain amount of effort from users. Therefore, managing numerical parameters can be a mundane and time-consuming activity [15, 16].

To mitigate this problem, a considerable number of studies have proposed various approaches to automatic selection of synthesis parameters [16-28]. Such solutions allow musicians to define desired sounds in a more intuitive way (e.g., using timbral attributes [23-28] like *bright* and *hollow*) and automatically select parameters as inputs to sound synthesizers so that the synthesized sound satisfies the given description.

Besides parameter selection, there is also a concept of so-called feature synthesizers which are capable of producing sounds directly from a given set of audio features that are either provided by the user or extracted from a target sound [29-33].

1.3 Deep Convolutional Oscillator

As a solution to innovative and flexible waveform synthesis with intuitive parameters that can easily replace or augment wavetable oscillators, this research proposes a generative deep learning network with a decoder architecture. It has been designed to map low-dimensional vectors of descriptors given as inputs to the network to single-cycle waveforms in its output layer. These descriptors serve as controllable parameters, so they have been intentionally selected to have an intuitive relationship with the desired timbre of the synthesized sound. Outputs of the network are waveforms of a fixed duration which can be upsampled or downsampled for achieving desired pitch, as it is the case with a traditional wavetable oscillator. Such raw synthesized signal is intended to pass through further modifiers like wave shapers, amplifiers, filters, and audio effects.

The expected advantages of the taken approach in comparison to traditional wavetable oscillators are 1) the ability of producing a wide variety of waveforms instead of just reproducing and combining pre-existing ones, and 2) the use of more intuitive synthesis parameters. To achieve these benefits, we propose a deep learning architecture based on convolutions and an appropriate loss function.

2. METHODOLOGY

A prerequisite for building a convolutional oscillator with intuitive parameters was to choose an appropriate set of those input parameters. Each parameter is an attribute with a belonging numerical value from 0 to 1. They are intended to allow musicians to specify the desired timbre by setting numerical values (e.g., *bright* to 0.4, *nasal* to 0.1, and *harsh* to 0.3) when using the oscillator.

When choosing appropriate attributes, it was important to consider two requirements. The first requirement is that attribute should bear acoustical meaning and thereby allow intuitive, yet expressive control over the resulting timbre. The second requirement was that parameters can be calculated from the audio signal they define. In other words, parameters should be related to numerical audio features. This is an opposite direction to the intended use of the parameters, but this requirement entails an important positive consequence. This way, we can calculate ideal values of parameters for training example and thereby form the training set without manual labeling. In addition, having such objective target values, they can be used for consistent evaluation of results. To find parameters which satisfy both the aforementioned criteria, we reviewed the existing literature searching for audio features with proven relations to timbral semantic scales. For selected features developed feature extractors adjusted to the use with single-cycle waveforms.

The next step was constructing a well-distributed dataset for training and testing of the deep learning model. A set of waveforms was extracted from NSynth [10], a widely used dataset which contains a large number of annotated tones systematically recorded from commercial sample libraries. The extraction algorithm ensured that selected portions of snippets from the dataset have characteristics of single-cycle waveforms. Along with each extracted waveforms we also stored their audio features calculated using custom feature extractors.

After the dataset construction, the third step was to prepare multiple neural networks with different architectures and hyperparameters. We opted with four different versions with convolutional and upsamling layers, inspired by the architecture of a convolutional decoder. We defined and implemented a custom loss function, as well as a specific scoring measure used for validation. After training all variants with the training dataset, we used validation sets to compare the models and select the best one. Finally, we evaluated the winning model using the test data and qualitatively analyzed their generative capability.

3. INPUT PARAMETERS

Semantic scales for describing musical timbres of steady tones have been a subject of scientific research for more than a century [34, 35]. Using dimensionality reduction methods such as cluster analysis [35-37], principal component analysis [39-41], and factor analysis [42], researchers have proposed concise sets of interpretable attributes for describing musical timbres. The most common conceptual labels in the examined papers were related to luminance, texture, temperature, and width. These dimensions served as a starting point for choosing attributes for controlling the generative oscillator.

Before fixing the set of attributes, we considered audio features which related to the most common semantic scales by following the directions from the previous studies [43, 44] which also overlap with practices in more recent studies on generative synthesis [45] and oscillator theory [46]. The limitation of working with repeated single-cycle waveforms is the lack of temporal changes longer than one cycle which excluded audio features such as spectral flux and inharmonicity. Based on relevance of semantic scales and appropriateness of corresponding audio features, we finally opted for three attributes: *bright* (luminance), *warm* (temperature), and *rich* (width). These attributes are not

sufficient to describe more complex and evolving sounds, but it is not their purpose anyway. The attributes are used only as inputs to the generative oscillator and can be observed as a substitution to wavetable and waveform indices in traditional wavetable oscillators.

The audio features selected in relationship to the attributes were spectral centroid, relative energy of odd-numbered partials, and spectral spread. The spectral centroid represents the gravity center of a frequency spectrum and therefore indicates the relative balance of low- and high-frequency energy. This feature is correlated with the impression of brightness [47]. The relative strength of odd-numbered partials is calculated as a ratio between the summed magnitudes of odd-numbered partials and the sum of all partials. It correlates with the perception of fullness or warmth of the sound [48]. Finally, the spectral spread is the standard deviation of the signal spectrum which indicates pureness or richness of the sound [49].

To extract these features from single-cycles waveforms, the algorithm first concatenates six repetitions of the waveform to increase the resolution of the discrete spectrum, applies the Hanning window, calculates the Fast Fourier Transform, and extracts audio features. As audio features can have different ranges of values and some can be perceived logarithmically, we transform them to fit the range [0, 1] and to achieve a linear relation to sound perception following the same approach as in similar studies [24, 50].

4. DATASET PREPARATION

Waveforms for training, validation and testing of deep learning models were extracted from NSynth dataset [10] which contains 305979 four-second, monophonic snippets taken at the sampling frequency of 16 kHz. Each snippet encompasses a tone recorded from one of 1006 instruments with a known MIDI velocity (one of five predefined levels) and a known MIDI pitch (between 21 and 108). To achieve the same duration of waveforms in the dataset, we selected snippets with MIDI note 31 (G1) from all available instruments and velocities. We opted for such a lower note to have a longer waveform size. This helps with restraining interpolation errors that may happen when the synthesized waveform is reproduced at a frequency different than its original frequency.

To determine positions of single-cycle waveforms to be extracted from snippets, the algorithm does two steps. First, it selects five random positions between timestamps of 0.5 and 2 seconds to avoid transients at the beginning of the tone and decay phase after its release. Then it uses each of random positions as the starting point for searching for the sample with the minimal absolute amplitude within next 327 samples (to encompass one full cycle for G1 at 16 kHz). This way we ensure that each waveform starts with a sample with a value close to zero. The sample with the lowest absolute amplitude is taken as the starting sample of a waveform. The periodicity is confirmed by calculating linear correlation with the neighboring blocks of samples to ensure that each waveform was taken from the steady part of the tone. The next step was the interpolation used to raise the fundamental frequency from 49 Hz to 50 Hz so that the cycle length becomes an integer number at the given sampling frequency of 16 kHz, specifically, 320 samples pre cycle. Waveforms with all zero values were removed from the dataset.

We split the extracted waveforms from the into the training, validation, and testing sets following the categorization of the corresponding snippets in NSynth dataset. After removing empty waveforms, this process resulted with 14430, 736, and 232 waveforms in the training, validation, and testing sets, respectively. For each waveform, we calculated and stored their feature vector as described in the previous section. Figure 1 shows three randomly selected waveforms and their feature vector.

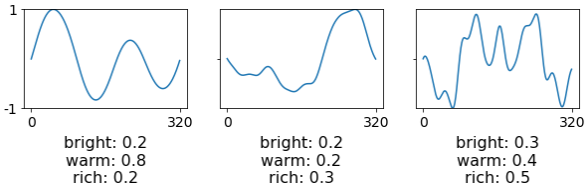


Figure 1. Three randomly selected waveforms from the training set with their attributes obtained from the extracted spectral audio features.

5. CONVOLUTIONAL DECODER

5.1 Architecture

The purpose of the deep neural network proposed within this research is to produce raw single-cycle waveforms based on given input parameters. Therefore, the network has 3 inputs and up to 318 outputs which represent individual samples of the waveform with possible values between -1 and 1. Since it maps a low-dimensional input vector to an output of a higher dimensionality, the network's structure was inspired by decoders [51].

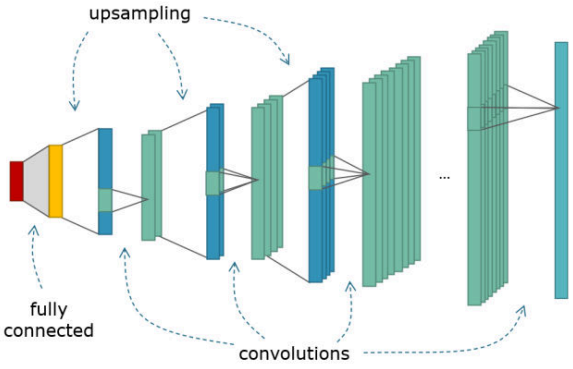


Figure 2. Architecture of the deep convolutional waveform generator.

The first hidden layer was fully connected with the inputs, while the rest of the network was constructed by stacking multiple pairs of upsampling and convolutional layers on top of each other. Convolutional layers turned to be very efficient in handling temporal dependencies in audio signals [6] and musical sequences [1]. Unlike their typical use in obtaining more condensed representations, here we deal with a reverse problem – mapping a low-dimensional representation to a full signal. For that reason, we

altered upsampling layers with convolutional layers (with short kernels) multiple times (Figure 2) so that the network can gradually, through multiple layers, capture dependencies on various time spans within the waveform.

The fully connected layer had 22 neurons, while each upsampling layer doubled the length in comparison to the previous layer. After the last convolutional layer from the aforementioned stack, we added an additional convolutional layer without upsampling with only one convolutional filter and with a slightly longer kernel to smooth the waveform. All the layers used the hyperbolic tangent activation.

To avoid artifacts in the output waveform that could appear as a consequence of padding in the convolutional layers, we decided to avoid any padding. A negative side effect is that sizes of convolutional kernels affect sizes of subsequent convolutional layers and eventually the size of the final output of the network. However, this problem manifests only during the exploration phase in which we validate various kernels. Once when we opt for our final model, the problem is no longer relevant.

In order to test different variations based on the described architecture, we constructed, trained, and validated four models with different numbers of filters and kernel sizes in convolutional layers as shown in Table 1.

Model	Number of filters	Kernel sizes
Model 1	2, 4, 8, 16, 16, 1	2, 2, 2, 2, 2, 4
Model 2	8, 16, 32, 64, 64, 1	2, 2, 2, 2, 2, 4
Model 3	2, 4, 8, 16, 16, 1	4, 4, 4, 4, 4, 8
Model 4	8, 16, 32, 64, 64, 1	4, 4, 4, 4, 4, 8

Table 1. For each model, the corresponding row shows the number of filters and kernel sizes per convolutional layer (starting from the one closest to the input of the network).

5.2 Loss Function

Audio features do not uniquely describe waveforms. Radically different waveforms can have similar or even same audio spectral features (e.g., waveforms with shifted phases). For that reason, input parameters of the convolutional oscillator do not uniquely define the exact waveform on the output of the network. Instead, various outputs are acceptable, as long as they satisfy the input description to some extent. For example, if the attribute *bright* is set to 0.9, the user expects any waveform that with a very high spectral centroid, and not one specific waveform. This phenomenon affected the selection of the loss function, as we could not simply use a regression loss calculated by sample-wise comparison between the training waveform and the output waveform.

Instead, we opted for the mean squared error between their normalized power spectra. Discrete Fourier Transform is differentiable which makes it appropriate for a loss function that allows training with a gradient descent. Also, by using the full spectra instead of the three selected spectral features, the network has a chance to learn how to produce waveform with the spectral distribution similar to one from the training set.

To get the power spectrum, the algorithm concatenates six repetitions of the waveform to increase the resolution, applies the Hanning window, calculates the Fast Fourier Transform, and calculates the normalized power spectrum. The loss function returns the mean squared error between corresponding discrete frequency components of these two normalized power spectra.

5.3 Training

The training of all four models was conducted using 14430 waveforms from the training set and their audio features as inputs to the network. We employed the Adam optimizer with the learning rate of 0.001. The batch size was 64, while the number of epochs was 20 to capture the convergence of the loss during the training process. The training time on Intel® Core™ i5-8250U CPU@1.60 GHz using TensorFlow 2.7.0 took between 107 seconds per epoch on average for Model 1 and 119 seconds per epoch on average for Model 4.

5.4 Validation

For validation and testing we used a scoring metric different from the loss function. The purpose of this metric was to indicate how well waveforms produced by the neural network satisfy the input parameters. For that reason, we opted for a mean absolute error (MAE) between the input parameters and the corresponding audio features of the synthesized waveform. It is important to note that waveforms from the validation set were used only to obtain their audio features so that we can validate the models based on realistic distribution of the features.

All models performed well on the validation set with regards to the selected metric, suggesting that they did not overfit to the training data. Table 2 shows the MAE per attribute per model. Based on these results, we selected Model 1 for testing and further analysis.

Model	Bright	Warm	Rich
Model 1	0.08	0.07	0.19
Model 2	0.14	0.08	0.29
Model 3	0.14	0.07	0.3
Model 4	0.15	0.07	0.28

Table 2. Mean absolute error between the input parameters and features of the output waveforms per attribute for each trained model.

6. RESULTS

6.1 Metrics

After selecting Model 1, it was evaluated using features of 232 waveforms from the testing dataset which had not been previously used neither for training, nor for the model selection. The mean absolute errors obtained on the testing

set were 0.07, 0.05, and 0.18 for attributes *bright*, *warm*, and *rich* respectively.

While the presented MAE indicates how close the spectral characteristics of the generated waveforms are to the input requirement, we were also interested to which extent these characteristics corelate to magnitudes of input attributes, so the Pearson correlation coefficient was calculated for each attribute. As expected, strong or very strong positive correlations occurred: 0.77 for *bright*, 0.97 for *warm*, and 0.43 for *rich*. Such results can be interpreted in a way that by increasing the parameters for brightness, warmth, and richness, the generated waveform becomes expectedly brighter, warmer, and richer.

The inference time was measured on Intel® Core™ i5-8250U CPU@1.60 GHz using TensorFlow 2.7.0. An inference step took on average 86 µs, which is an order of magnitude faster than what it takes to reproduce a waveform of 320 samples at the sampling frequency of 16 kHz. This finding confirms that the generative convolutional oscillator works sufficiently fast to replace a traditional wavetable oscillator in synthesizers running on PC platforms.

6.2 Generative Capabilities

To analyze the generative capabilities of the convolutional oscillator, we uniformly sampled the whole space of input parameters by iterating through 20 equidistant points for each parameter. This process resulted with the total of 8000 triples of parameters which we used as inputs to the generative model in order to analyze the distribution of its outputs.

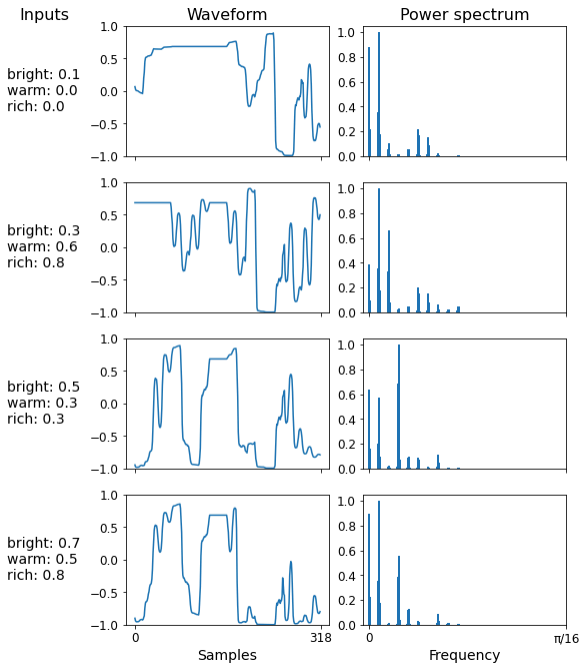


Figure 3. Columns from left to right: 1) values of input parameters, 2) generated waveforms, and 3) first 500 Hz of their normalized power spectra (obtained from the waveforms repeated for multiple times at the constant, original frequency).

By observing generated waveforms, we found an interesting pattern. Larger sudden changes in neighboring samples most often happen in several specific positions within the waveform. This is most likely the consequence of the network architecture with upsampling layers and convolutions with short kernels. Besides these patterns, the diversity of the generated waveform shapes was significant. Figure 3 shows four selected waveforms generated from the given inputs along with relevant parts of their power spectra. The spectra were obtained by repeating the waveforms at the constant, original frequency, so its harmonics expectedly lie on the same positions, but have different magnitudes. The sound examples available online (https://tinyurl.com/DeepConvOsc) demonstrate multiple steady tones (including those from Figure 3) and several swipes through various input parameters.

7. CONCLUSIONS AND FUTURE WORK

As a solution to generating a wide range of waveforms based on an intuitive input description, this paper proposes a novel architecture for a deep learning network. Even though the network has never received real-world waveforms as inputs or outputs during the learning process, it managed to learn how to produce viable waveforms. The generated waveforms generally match the input parameters well when the parameters are drawn from a realistic distribution, as it was the case with our training, validation, and testing data sets. While the results on the validation and test sets are very promising, the future work should investigate how the network will perform on inputs that are not drawn form a real-world distribution. The performance on arbitrary input parameters is expected to drop, since the network has been trained on audio features extracted from real signals in which some interdependencies inherently occurred.

The fact that the network managed to learn a complex mapping from a very low-dimensional space to the space of full single-cycle waveforms is due to its architecture. Alternations between upsampling and convolutional layers allow the network to construct waveforms taking into account their global and local structure. An idea for the future work is to consider further improvements of the architecture so that the network can directly benefit from thousands of real waveforms available in the training set. For example, such an architecture could be based on a convolutional variational autoencoder with additional modifications for connecting the input parameters with the latent space.

In conclusion, the convolutional generative network presented in this paper seems to have satisfactory generative capability. Due to its small size and computational efficiency, such a network can easily replace a traditional wavetable oscillator. While the notion of a convolutional generative network for waveform synthesis semes promising, further research is needed to investigate and possible improve its performance on arbitrary inputs.

8. REFERENCES

[1] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," arXiv preprint arXiv:1703.10847, 2017.

[2] H.-W. Dong, et al., "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in Proceedings of the Thirty-second AAAI conference on artificial intelligence, 2018.

[3] A. Shaw, *MusicAutobot*, Jan. 2020, Accessed on: Jan 16, 2020. Available: https://github.com/bearpelican/musicautobot, 2020.

[4] C.-Z. A. Huang, et al., "Music transformer," arXiv preprint arXiv:1809.04281, 2018.

[5] S. Mehri, et al., "SampleRNN: An unconditional end-to-end neural audio generation model," arXiv preprint arXiv:1612.07837, 2016.

[6] A. van den Oord, et al., "Wavenet: A generative model for raw audio," arXiv preprint arXiv:1609.03499, 2016.

[7] S. Vasquez, and M. Lewis, "Melnet: A generative model for audio in the frequency domain," arXiv preprint arXiv:1906.01083, 2019.

[8] J. Engel, et al., "GANSynth: Adversarial neural audio synthesis," arXiv preprint arXiv:1902.08710, 2019.

[9] A. Natsiou, and S. O'Leary, "Audio representations for deep learning in sound synthesis: A review," arXiv preprint arXiv:2201.02490, 2022.

[10] Engel, Jesse, et al., "Neural audio synthesis of musical notes with WaveNet autoencoders," in the Proceedings of the International Conference on Machine Learning, PMLR, 2017, pp. 1068-1077.

[11] U. Andresen, "A new way in sound synthesis," in Proceedings of the Audio Engineering Society Convention 62, 1979.

[12] J. O. Smith III, "Viewpoints on the history of digital synthesis," in Proceedings of the International Computer Music Conference, 1991.

[13] S. Shan, L. Hantrakul, J. Chen, M. Avent, D. Trevelyan, "Differentiable Wavetable Synthesis," in arXiv preprint arXiv:2111.10003, 2021.

[14] G. Kreković, "A concept of a wavetable oscillator based on a neural autoencoder," in Proceedings of Audio Mostly, 2021, pp. 240-243.

[15] G. Kreković, "Insights in habits and attitudes regarding programming sound synthesizers: a quantitative study," the Proceedings of the 16th Sound and Music Computing Conference, 2019, pp. 316-322.

[16] E. Miranda, "An artificial intelligence approach to sound design," in Computer Music Journal, vol. 19, no. 2, 1995, pp. 59–75.

[17] S. Fasciani, "TSAM : a tool for analyzing , modeling, and mapping the timbre of sound synthesizers," in Proceedings of 13th Sound and Music Computing Conference, 2016, pp. 129–136.

[18] A. Horner, J. Beauchamp, and L. Haken, "Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis," in Computer Music Journal, vol. 17, no. 4, 1993, pp. 17–29.

[19] M. J. Yee-King and M. Roth, "SynthBot – An Unsupervised Software Synthesizer Programmer," in Proceedings of the International Computer Music Conference, 2008.

[20] M. Chinen and N. Osaka, "Genesynth: Noise Band-Based Genetic Algorithm Analysis/Synthesis Framework," in Proceedings of the International Computer Music Conference, 2007.

[21] M. Macret, "Automatic tuning of the OP-1 synthesizer using a multi-objective genetic algorithm," in Proceedings of the Sound and Music Conference, 2013, pp. 614–621.

[22] M. J. Yee-King, L. Fedden, and M. d'Inverno, "Automatic Programming of VST Sound Synthesizers Using Deep Networks and Other Techniques," in IEEE Transactions on Emerging Topics in Computational Intelligence, Vol. 2, No. 2, 2018, pp. 150–159.

[23] A. Gounaropoulos and C. Johnson, "Synthesising Timbres and Timbre Changes from Adjectives/Adverbs," in P. Collet et al. Applications of Evolutionary Computing, Springer-Verlag, Berlin, 2006.

[24] G. Kreković, A. Pošćić, D. Petrinović, "An Algorithm for Controlling Arbitrary Sound Synthesizers Using Adjectives," in Journal of New Music Research, Vol. 4, No. 45, 2016, pp. 375–390.

[25] R. Ethington and B. Punch, "SeaWave: A System for Musical Timbre Description," in Computer Music Journal, Vol. 18, No. 1, 1994, pp. 30–39.

[26] A. Pošćić and G. Kreković, "Controlling a Sound Synthesizer Using Timbral Attributes," in Proceedings of the Sound and Music Computing Conference, Stockholm, 2013, pp. 467–472.

[27] R. Clement, "Automatic Synthesiser Programming," in Proceedings of the International Computer Music Conference, Universitiy of Huddersfield, UK, 2011, pp. 155–158.

[28] G. Kreković and D. Petrinović, "Intelligent Exploration of Sound Spaces Using Decision Trees and Evolutionary Approach," in Proceedings of the International Computer Music Conference joint with

Sound and Music Computing Conference, Athens, Greece, 2014, pp. 1263–1270.

[29] D. Wessel, "Timbre Space as a Musical Control Structure," in Computer Music Journal, Vol. 3, No. 2, 1979, pp. 45–52.

[30] C. Hourdin, G. Charbonneau, and T. Moussa, "A Sound Synthesis Technique Based on Multidimensional Scaling of Spectra," in Computer Music Journal, Vol. 21, No. 2, 1997, pp. 40–55.

[31] T. Jehan, "Perceptual Synthesis Engine: An Audio-Driven Timbre Generator," Master thesis, Massachusetts Institute of Technology, Cambridge, 2001.

[32] J. W. Beauchamp, The Sound of Music: Analysis, Synthesis, and Perception of Musical Sounds, Springer, New York, 2007.

[33] D. Mintz, "Towards Timbral Synthesis: A New Method for Synthesizing Sound Based on Timbre Description Schemes," Master thesis, University of California, Santa Barbara, 2007.

[34] H. L. von Helmholtz, On the Sensations of Tone as a Physiological Basis for the Theory of Music. Cambridge University Press, 2009. (Original work published 1877).

[35] W. H. Lichte, "Attributes of complex tones," in Journal of Experimental Psychology, vol. 28, no. 6, 1931.

[36] R. A. Kendall, and E. C. Carterette, "Verbal attributes of simultaneous wind instrument timbres: Von Bismarck adjectives," in Music Perception, vol. 10, no. 4, 1993, pp. 445–468.

[37] A. C. Disley, D. M. Howard, and A. D. Hunt, "Timbral description of musical instruments," in Proceedings of the International Conference of Music Perception and Cognition, 2006, pp. 61–68.

[38] A. Zacharakis, K. Pastiadis, J. D. Reiss, and G. Papadelis, "Analysis of musical timbre semantics through metric and non-metric data reduction techniques," in Proceedings of the International Conference of Music Perception and Cognition, 2012, pp. 1177–1182.

[39] G. von Bismarck, "Sharpness as an attribute of the timbre of steady sounds", in Acustica, no. 30, 1974, pp. 159–192.

[40] R. A. Kendall, E. C. Carterette, and J. M. Hajda, "Perceptual and acoustical features of natural and synthetic orchestral instrument tones," in Music Perception, no. 16, 1999, pp. 327–364.

[41] T. Lokki, J. Pätynen, A. Kuusinen, H. Vertanen, and S. Tervo, "Concert hall acoustic assessment with individually elicited attributes," in Journal of theAcoustical Society of America, 130, 2011, pp. 845–849.

[42] V. Alluri and P. Toiviainen, "Exploring perceptual and acoustical correlates of polyphonic timbre," in Music Perception, vol. 27, no. 3, 2010., pp. 223–242.

[43] P. Herrera-Boyer, G. Peeters, and S. Dubnov, "Automtaic classification of musical instrument sounds," in Journal of New Music Research, vol. 32, no. 1, 2003, pp. 3–21.

[44] S. Donnadieu, "Mental representation of the timbre of complex sounds," in J. W. Beauchamp (Ed.), Analysis, synthesis, and perception of musical sounds, New York, Springer, 2007, pp. 272–319.

[45] J. Nistal, S. Lattner, and G. Richard, "DrumGAN: Synthesis of drum sounds with timbral feature conditioning using Generative Adversarial Networks," arXiv preprint arXiv:2008.12073, 2020.

[46] G. Essl, "Iterative phase functions on the circle and their projections: Connecting circle maps, waveshaping, and phase modulation," in International Symposium on Computer Music Multidisciplinary Research, 2019.

[47] J. M. Grey, and J. W. Gordon, "Perceptual effects of spectral modifications on musical timbres," Journal of the Acoustical Society of America, vol. 63, no. 5, 1978, pp. 1493–1500.

[48] K. Jensen, Timbre models of musical sounds, PhD thesis. Department of Computer Science, University of Copenhagen, Copenhagen, Denmark, 1999.

[49] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," Technical Report, IRCAM: Paris, France, 2004

[50] J. McDermott, Evolutionary computation applied to the control of sound synthesis, PhD thesis. University of Limerick, Ireland, 2008.

[51] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," in AIChE journal, vol. 37, no. 2, 1991, pp. 233-243.

LEARNING SPARSE ANALYTIC FILTERS FOR PIANO TRANSCRIPTION

Frank Cwitkowitz, Mojtaba Heydari, Zhiyao Duan
Audio Information Research Lab, University of Rochester
{fcwitkow,mheydari}@ur.rochester.edu zhiyao.duan@rochester.edu

ABSTRACT

In recent years, filterbank learning has become an increasingly popular strategy for various audio-related machine learning tasks. This is partly due to its ability to discover task-specific audio characteristics which can be leveraged in downstream processing. It is also a natural extension of the nearly ubiquitous deep learning methods employed to tackle a diverse array of audio applications. In this work, several variations of a frontend filterbank learning module are investigated for piano transcription, a challenging low-level music information retrieval task. We build upon a standard piano transcription model, modifying only the feature extraction stage. The filterbank module is designed such that its complex filters are unconstrained 1D convolutional kernels with long receptive fields. Additional variations employ the Hilbert transform to render the filters intrinsically analytic and apply variational dropout to promote filterbank sparsity. Transcription results are compared across all experiments, and we offer visualization and analysis of the filterbanks.

1. INTRODUCTION

Automatic music transcription (AMT) is an essential capability for intelligent systems which analyze music [1]. The task is part of the broader music information retrieval (MIR) class of machine learning problems. AMT seeks to retrieve all of the information necessary to develop a score which accurately represents the music. However, given this complexity, the problem is typically reduced to the aim of estimating all notes, where a note is characterized by its pitch, time of onset, and duration [1]. Multi-instrument AMT is challenging, leading many to develop algorithms targeted for successful single-instrument AMT as an initial goal. In particular, piano transcription has been an active research task in AMT, given the wide availability of note annotations [2], and the consistency of piano timbre compared to multi-instrument ensembles [3].

Most machine learning methods involving AMT, and MIR in general, begin with a feature extraction stage which calculates a 2D time-frequency representation (TFR) for a piece of audio. Common choices for this step include the Mel-spectrogram or the constant-Q transform (CQT). In AMT algorithms, the features must contain at least enough information to detect and track notes. While standard TFRs are viable for this purpose, they may not be optimal. In particular, most frequency analysis methods do not explicitly model note characteristics. They also require careful design choices like filter shapes and other hyperparameters. Furthermore, there is fundamentally no information gained in moving to the frequency-domain.

In this work, we focus on filterbank learning for piano transcription, extending the preliminary work of [4]. Filterbank learning is a way to circumvent the use of handcrafted features by replacing or augmenting the TFR calculation with the response from a bank of learnable filters. As the learnable filters are tuned jointly with a backend model for the task at hand, ideally they can model task- and domain-specific characteristics of the input signal.

Although filterbank learning is widely applicable to various audio processing problems, we target piano transcription because it is a complex task which, intuitively, may benefit from modeling notes in the time-domain. The filterbank learning paradigm provides an opportunity to model some of the more obscure properties of musical notes, such as onset or offset behavior, harmonic structure, inharmonicity, or more generally timbre. Piano transcription is also a task with plenty of annotated data and consistency, two characteristics which can simplify the problem.

We adopt the Onsets & Frames piano transcription model [2, 5] and replace the feature extraction stage with a learnable filterbank module [1]. We utilize a 1D convolutional structure for the filterbank, and experiment with several initialization strategies, model variations and regularization techniques. The filters represented in the module are complex, and can be employed with a standard hop size or stride. We compare the transcription results of each experiment, and provide an analysis of the learned filterbanks. We show that in general, the filters converge to sparse, unique shapes, which we speculate to be modeling various note characteristics. We also demonstrate that filterbanks initialized with random weights achieve comparable performance to those with fixed TFR initializations.

2. RELATED WORK

Filterbank learning has gained traction in recent years as a means to perform audio-related machine learning tasks in an end-to-end fashion. Some of the first filterbank learning attempts emerged to tackle problems within the speech

[1] All code is available at https://github.com/cwitkowitz/sparse-analytic-filters.

Copyright: © 2022 Frank Cwitkowitz et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

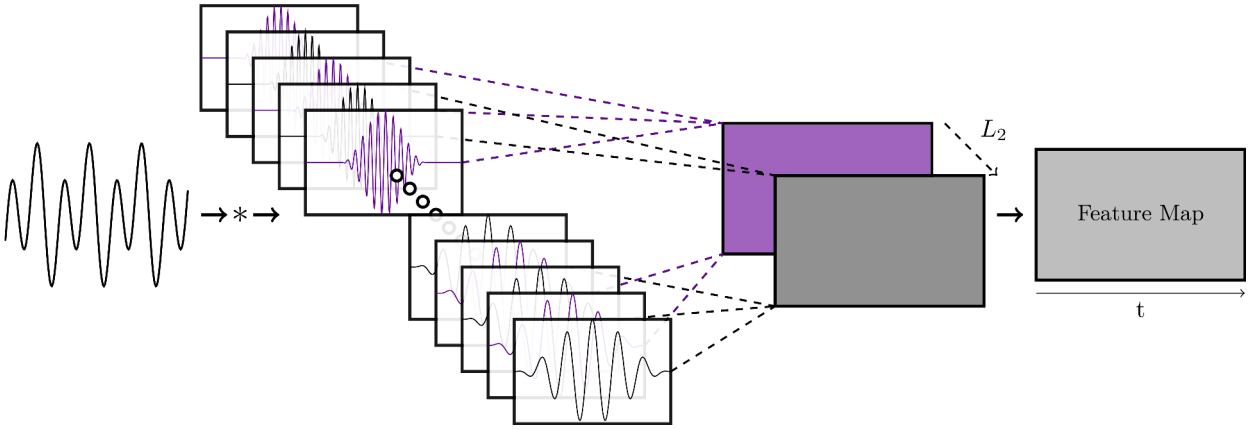


Figure 1. Complex filterbank learning module. The real (black) and imaginary (purple) part of each filter are learned independently. The real and imaginary responses are combined channel-wise using L_2 pooling to obtain a feature map.

community [6, 7], replacing the more widely used Mel-Frequency Cepstral Coefficients. This eventually lead to powerful models for tasks such as speech separation [8, 9].

Filterbank learning has also been applied as a frontend for music data. However, this has mainly been for high-level tasks such as music auto-tagging [10–12] or various classification problems [13]. For lower-level tasks, such as AMT, filterbank learning has received only some attention [14, 15]. Music data has rich characteristics, patterns, and relationships at many layers of abstraction, including the note-level, the instrument-level, etc. As such, filterbank learning carries significant potential for discovering, capturing, and leveraging task-specific information.

Many filterbank learning approaches learn strictly real-valued filters, and attain shift-invariance by pooling the response at small hops across time [6]. These filterbanks are analogous to standard 1D convolutional blocks in deep networks. In contrast, the filters used in fixed TFR calculations, e.g., CQT, are typically complex and analytic, which means that they implicitly encode phase. As such, they are stable with relatively large hop sizes, and there is no need to apply further temporal pooling on their responses.

Some filterbank learning approaches have extended the 1D convolutional approach to implement complex filters with grouped real-valued filters representing the real and imaginary part of the complex filter [16–18]. While the filters in these examples are complex, if their analytic property is not preserved, they will have an asymmetric frequency response about 0 Hz and no longer exhibit shift-invariance in the magnitude response. Moreover, the 1D convolutional layer approach sometimes suffers from too little constraints, leading to noisy filters with no distinct or localized frequency response. One way to address this is to add soft constraints, e.g., initializing the filterbank with the weights approximating a standard transform [6].

Recently, there has been a trend to further constrain filterbank learning [19, 20], by learning a small number of parameters which define fixed-shape filters. While these approaches may exhibit more stability, they can only really model simple bandpass filters and thus, lower the potential for discovering meaningful patterns in data at the signal level. Other approaches learn frequency-domain filters on top of spectrograms [21, 22], initializing them with harmonic relationships. We wish to discover these relationships naturally, rather than imposing them as a constraint.

Many approaches to piano transcription have attempted to realize a framework which can learn task-specific note characteristics. Non-Negative Matrix Factorization (NMF) has been proposed to learn properties such as the harmonic relationships, temporal evolution, attack, and decay of piano notes from a TFR [23, 24]. Convolutional sparse coding is a similar approach which operates in the time-domain, and was proposed as a way to estimate the activation of pre-acquired note impulse responses [3, 25].

More recently, deep neural networks (DNNs) have demonstrated success in learning to estimate discrete pitch activity from TFRs [2, 5, 26, 27]. Some approaches have attempted to design DNNs such that they naturally leverage information about note characteristics [28–31]. Since they are very powerful and efficient at learning features for many tasks, we hypothesize that DNNs can be utilized to learn better input features for acoustic models. Furthermore, we believe the proposed frontend will naturally learn to model note characteristics for piano transcription.

3. METHOD

The complex filterbank learning module is implemented as a 1D convolutional layer. It accepts a 1D signal as input and produces a real and imaginary feature map, which are combined using L_2 pooling, a simple mechanism for computing the magnitude. The output of the module is subsequently converted to log-amplitude and fed into a batch normalization layer. The filterbank is formulated such that an inner product is taken between a time-domain signal x and n_{bins} filters, indexed by μ, of respective lengths l_μ with weights θ_μ at discrete hops k spaced l_h samples apart:

$$X[k, \mu] = \sum_{n=0}^{l_\mu - 1} x[kl_h + n]\theta_\mu[n]. \quad (1)$$

Note that this operation is equivalent to convolution, or more precisely correlation, using a stride of l_h. The most

straightforward way to represent complex filters in this type of framework is to allocate two adjacent filters for the real and imaginary weights, and to combine their respective responses channel-wise using L_2 pooling [16]:

$$X[k,\mu] = \sqrt{(x * \mathcal{R}(\theta_\mu))^2 + (x * \mathcal{I}(\theta_\mu))^2}. \quad (2)$$

Note that with this representation, there are actually $2 \times n_{bins}$ filters to learn, and they are only implicitly associated via the subsequent pooling mechanism. Fig. 1 illustrates this filterbank architecture. While this approach can model complex filters, the filters are rarely analytic, unless they are initialized as analytic filters. This means that they may have non-zero responses to negative frequencies and may not be shift-invariant with respect to time.

Another way to model complex filters is to learn only the real part and to infer the imaginary part such that the complex filter is analytic [32]. This can be done using the Hilbert Transform, which computes the imaginary counterpart to a real signal, such that the resulting complex signal is analytic [2] . This can be expressed as

$$X[k,\mu] = \sqrt{(x * \mathcal{R}(\theta_\mu))^2 + (x * H(\mathcal{R}(\theta_\mu)))^2}, \quad (3)$$

where $H(\cdot)$ denotes the Hilbert transform. This variation learns shift-invariant filters with frequency responses containing only energy in the positive frequency range.

3.1 Initialization Strategy

Within the framework presented above, the weights θ_μ are initialized randomly by default. Without inserting any prior knowledge into the filterbank, it must learn to generate feature maps from scratch. This strategy has the potential to discover new filter shapes and characteristics that are not present in standard TFRs.

Alternatively, it is possible to insert weights into the filterbank such that it implements a time-domain variable-Q transform (VQT) if left untrained. The complex variable-Q response for filter μ can be computed in the time-domain by making the weights θ_μ complex basis functions with center frequency f_μ, sampling rate f_s, and smoothly varied Q-factor Q_μ:

$$\theta_{\mu,n} = w_{\mu,n} e^{-j\frac{2\pi f_\mu n}{f_s}}, \quad n = 0, \ldots, l_\mu. \quad (4)$$

The filter length $l_\mu = \left\lceil Q_\mu \frac{f_s}{f_\mu} \right\rceil$ is set such that the desired Q-factor, whether constant or smoothly varied, is maintained across all filters. The windowing function w is chosen as a Hanning window and matches the filter length l_μ for each filter. We avoid normalizing the response of each basis by filter length l_μ so that the weights of each filter are all comparable and responsive to a single learning rate. The center frequency of each filter in the VQT initialization is defined by selecting f_{min}, and applying

$$f_\mu = f_{min} \times 2^{\frac{\mu}{n_{bpo}}}, \quad (5)$$

where n_{bpo} is the number of bins or per octave.

The bandwidth of each respective filter is determined by $B_\mu = f_{\mu+1} - f_\mu + \gamma$. In a CQT, the ratio between the center frequency f_μ and the bandwidth B_μ, or the Q-factor, is constant, meaning that $\gamma = 0$. However, for large n_{bpo}, this can lead to filters with extremely tight bands, requiring long impulse responses. Modestly utilizing the constant bandwidth offset γ relaxes the constancy constraint in the lower frequency range and provides direct control over the size of the filter receptive fields, which must all be zero-padded to the largest filter in order to be stored in a 1D convolutional layer. We also use the VQT parameters to determine the receptive field size for random initialization.

Initializing the filterbank with the VQT weights provides a foundation to improve upon, and could even be thought of as pre-training the filterbank. Although there are multiple transforms one could utilize, the VQT is chosen because it is intuitively a better starting point for modeling note characteristics. Every $\frac{n_{bpo}}{12}^{th}$ filter is already locked onto a pitch in the Western music scale. Since we are primarily concerned with relationships between pitch and temporal characteristics for the AMT problem, we believe this is an appropriate initialization strategy.

An additional initialization strategy is the harmonic comb, where a set of harmonics $\mathcal{H}$ is defined, and separate filters with center frequency $f = h \times f_\mu$ are constructed for each $h \in \mathcal{H}$. All filters associated with a given μ are summed and collapsed into a single set of normalized weights for insertion into the 1D convolutional layer. In this way, each filter responds directly to the harmonics of its corresponding pitch, and characteristics such as inharmonicity or relative harmonic strength (timbre) can be fine-tuned.

3.2 Variational Dropout

Variational dropout [33] is a regularization technique which allows for the learning of sparse frontend filters with long receptive fields. It is similar to Gaussian dropout, except the dropout rate, or variance, corresponding to each weight is learned. During training we treat the weights as random variables, with their true values θ being the mean, and an additional set of learned parameters σ^2 being their variance. By learning the variance of each weight, variational dropout induces sparsity, as it pushes less important weights to have higher variance, and more important weights to have lower variance. This leads to more sparse and interpretable filters, and can deal with very long receptive fields effectively. The stochasticity can also stimulate the filterbank to take interesting and unexpected shapes as it is being trained. During each forward pass, noise is sampled from the variance of the filterbank response and added to the mean response:

$$X[k,\mu] \sim \mathcal{N}(x * \theta_\mu, x^2 * \sigma_\mu^2). \quad (6)$$

The parameters σ_μ^2 are jointly trained with the filterbank parameters θ_μ, and their gradient is calculated using a very close approximation to the KL-divergence [33]. In practice, we iterate upon $\log \sigma^2$ for improved stability. It should be noted that we only add noise to the real part of the feature map when using the Hilbert filterbank variant.

[2] Note that the filters represented with this variation may only be approximately analytic, due to the limitations of discrete processing.

	MAESTRO			MAPS		
Experiment	**Frame F_1**	**Note-On F_1**	**Note-Off F_1**	**Frame F_1**	**Note-On F_1**	**Note-Off F_1**
mel	91.80*	95.95*	83.44*	81.40*	81.42*	59.15*
mel	**90.91**	**95.82**	**83.14**	**81.26**	83.86	**59.07**
cqt	90.79	95.29	82.30	77.46	82.35	52.18
vqt	90.18	94.74	80.51	80.26	83.42	55.34
fixed comb	87.59	92.19	75.09	80.30	**84.64**	57.22
cl+rnd	86.70	91.22	73.72	70.03	78.77	43.92
cl+vqt	87.53	92.24	75.64	72.90	80.90	48.06
hb+rnd	86.50	91.30	73.19	76.13	80.00	51.88
hb+vqt	**87.81**	**92.63**	**76.01**	75.11	80.71	50.66
hb+rnd+brn	85.23	90.19	70.75	74.81	79.48	50.57
hb+rnd+gau	85.63	90.41	71.81	75.58	80.44	51.28
hb+rnd+var	86.04	90.61	72.27	73.59	80.09	47.99
hb+vqt+var	87.45	92.39	74.92	75.62	80.80	50.55
hb+comb+var	87.52	92.27	75.69	**76.74**	**80.98**	**52.57**

Table 1. Evaluation results for all baseline and filterbank experiments (separated by break). Filterbank learning experiment names are formatted in terms of variant (classic (*cl*) / Hilbert (*hb*)) + initialization (random (*rnd*) / *vqt* / *comb*) + dropout (none / Bernoulli (*brn*) / Gaussian (*gau*) / variational (*var*)). Results obtained using MAESTRO V_1 are indicated by *.

4. EXPERIMENTS

4.1 Model

The filterbank learning module is used as a frontend replacement for the feature extraction stage of the Onsets & Frames model [2,5] and jointly trained for the task of piano transcription. While there have been more recent models with improved performance, *e.g.* [31, 34], we stick with Onsets & Frames as a baseline due to its simplicity for the purposes of filterbank analysis. The original model takes as input a Mel spectrogram and produces a frame-level salience estimate for all note pitches, independently. Note predictions are generated using the simple post-processing steps proposed in the original paper, except we do not use the note predictions to refine the salience estimate, nor do we perform the additional velocity estimation. Our experimental setup is the same as the original paper [5] with the subsequent improvements proposed in [2]. We utilize the same hyperparameters, running each experiment for 2000 iterations, where each music piece is accessed once per iteration to sample a sequence of frames for a batch.

4.2 Datasets

MAESTRO [2] is used for training, validation, and testing. Specifically, we use version 3 (V_3) of the dataset for all but one experiment, which is run on version 1 (V_1). We follow the suggested partitioning for both versions. We also evaluate on the acoustic partitions (ENSTDkAm/ENSTDkCl) of MAPS [35], to inspect the generalization potential of our method. We downsample all audio to $f_s = 16$ kHz, and use a hop length of $L_h = 512$ samples.

4.3 Metrics

We evaluate experiments using the same transcription metrics as in [5], which include frame-wise, note-wise, and note-wise with offset scores. Each of these metrics is calculated in terms of precision, recall, and f1-score, using *mir_eval* [36]. For frame-wise evaluation, detected pitches are compared to the ground-truth at the frame-level. For note-wise evaluation, note estimates for an entire piece are compared to the ground-truth. A note is considered correct if its pitch is within half a semitone of the true value, and if its onset is within 50 ms of the true value. The note-wise with offset metric additionally compares the estimated offset to the true value, a correct estimate being within 50 ms or 20% of the ground truth duration, whichever is larger.

4.4 Results

We conduct several experiments to verify our implementation of Onsets & Frames, and to assess the strength of various features. We use $n_{bins} = 229$ and HTK frequency spacing for Mel spectrogram, as in [2]. We choose $f_{min} \approx 32.7$ Hz, $n_{bins} = 252$ and $n_{bpo} = 36$, or 3 bins per semitone for CQT and VQT. We calculate $\gamma = \frac{24.7}{0.108 * Q}$, such that all filters have a bandwidth proportional to the ERB scale by a constant factor [37] for the VQT experiment. We also experiment with an untrained filterbank initialized using the harmonic comb strategy ($\mathcal{H} = \{1, ..., 5\}$). Results for these experiments are provided at the top of Table 1. We observe that Mel spectrogram performs best in every metric besides note-onsets for MAPS, where the untrained comb filterbank slightly outperforms it. Interestingly, the comb filterbank is inferior in all other metrics.

Next we conduct several filterbank learning experiments where the output of the filterbank is fed to the model instead of a TFR. The same VQT parameters used in the baseline experiments are used to initialize the filterbanks. We vary the filterbank variant, initialization, and regularization method across all experiments. For Gaussian and variational dropout, we initialize all $\log \sigma^2_{\mu,n} = -10$, and for Bernoulli dropout we use a dropout rate of 10%. For variational dropout, the KL-divergence term is scaled by a factor of 0.01, such that the implicit sparsity objective does not dominate the training process. Results for these filterbank experiments are provided at the bottom of Table 1.

The Hilbert transform-based architecture performs similarly to the classic framework for random initialization, but

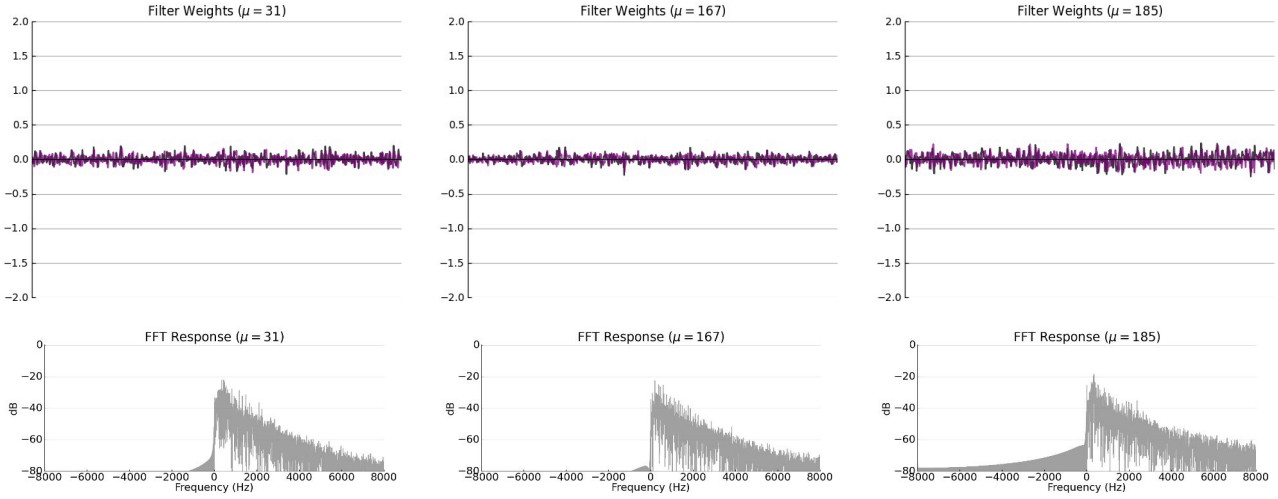


Figure 2. Examples of filters from the *hb+rnd+var* experiment which exhibit a high degree of sparsity. These filters have small weights and as such likely have little relevance to the downstream model.

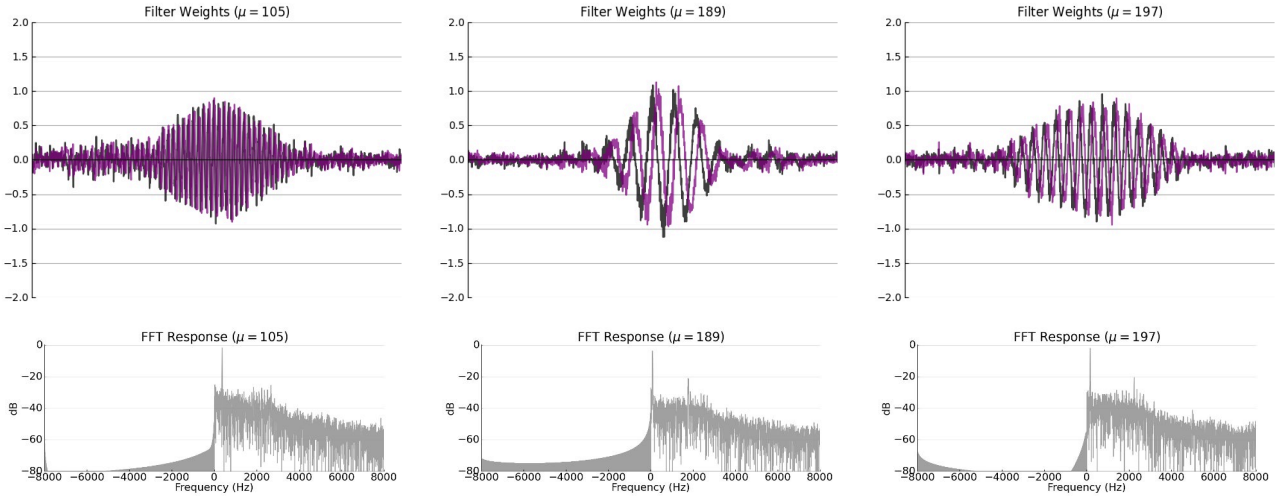


Figure 3. Examples of filters from the *hb+rnd+var* experiment which are well-localized within the receptive field and exhibit coherent shapes in both the time- and frequency-domain.

performs slightly better for VQT initialization, which tends to slightly outperform the random initialization. The variational dropout regularization method performs the best among regularization methods on MAESTRO, though still under experiments with no regularization. None of the learned filterbanks outperform the fixed TFRs. Nonetheless, they demonstrate reasonable performance, even when initialized randomly. The learned filterbanks, however, do not seem to generalize very well to the MAPS dataset. In terms of generalization to MAPS, the harmonic comb initialization does the best, and variational dropout generally underperforms other regularization methods.

5. DISCUSSION

We provide visualization of a few examples from the filterbank learning experiment corresponding to the Hilbert variant with random initialization and variational dropout (*hb+rnd+var*), which we consider to have produced the most interesting filters. These are presented in Fig. 2-6. In each figure, the time-domain visualization for three filters is displayed in the top row, with the corresponding frequency-domain visualization for each respective filter in the bottom row. The specific examples are chosen because they appear to belong to the same category of learned filter.

In general, the filters learned for the *hb+rnd+var* experiment tend to exhibit a high degree of sparsity, with coherent shapes in both the time- and frequency-domain. Fig. 2 and Fig. 3 illustrate these properties, respectively. The non-zero filter weights tend to be well-localized within the receptive field, with weights closer to zero being closer to the edges and more prominent weights being closer to the center. The tendency of the filters to be sparse is highly desirable, as it encourages more modular filters and leads

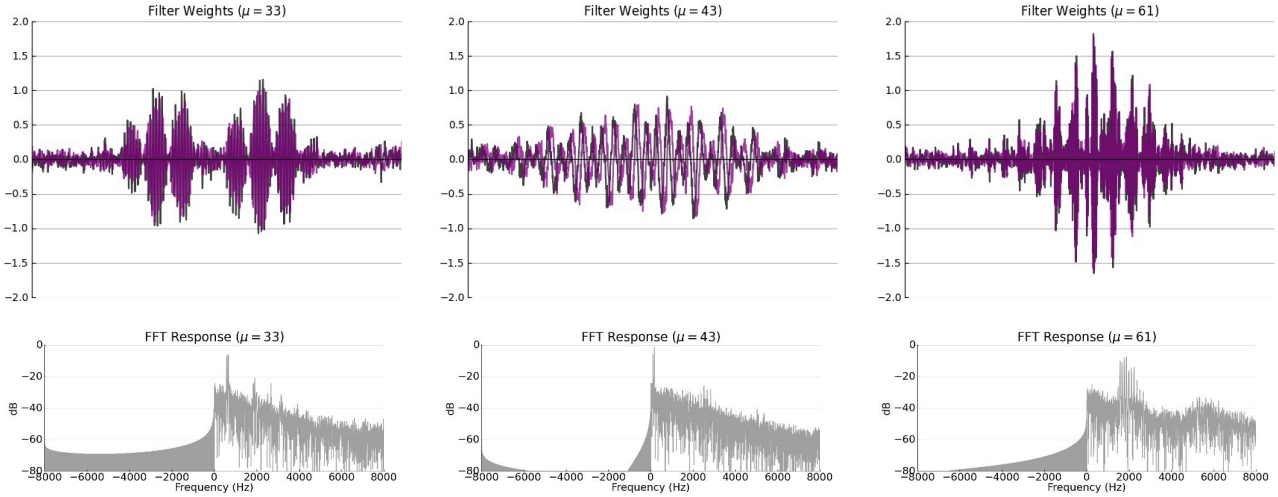


Figure 4. Examples of filters from the *hb+rnd+var* experiment which support multiple frequencies.

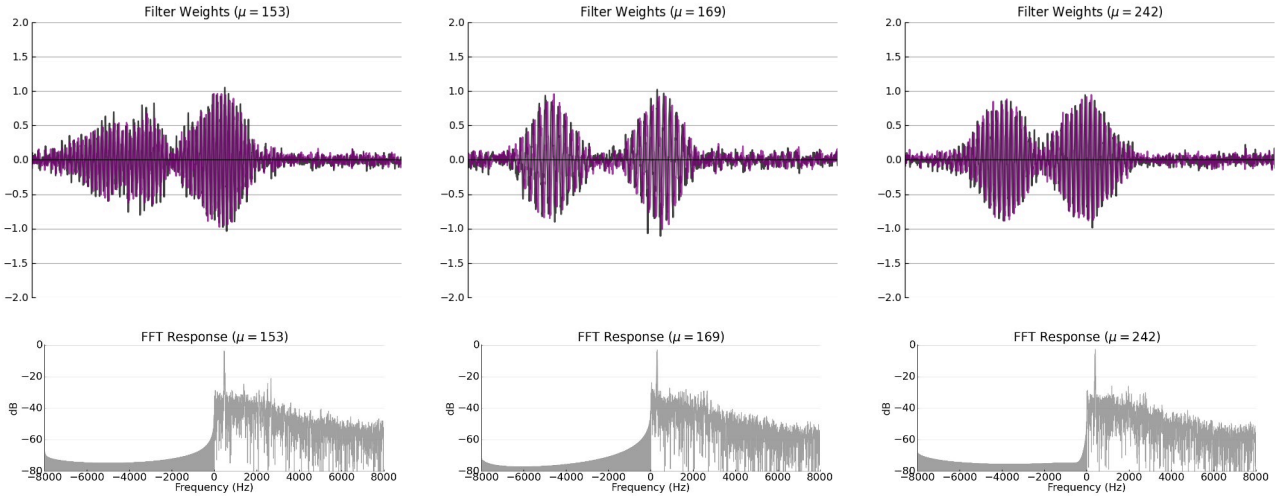


Figure 5. Examples of near-symmetric filters from the *hb+rnd+var* experiment comprising two impulse lobes.

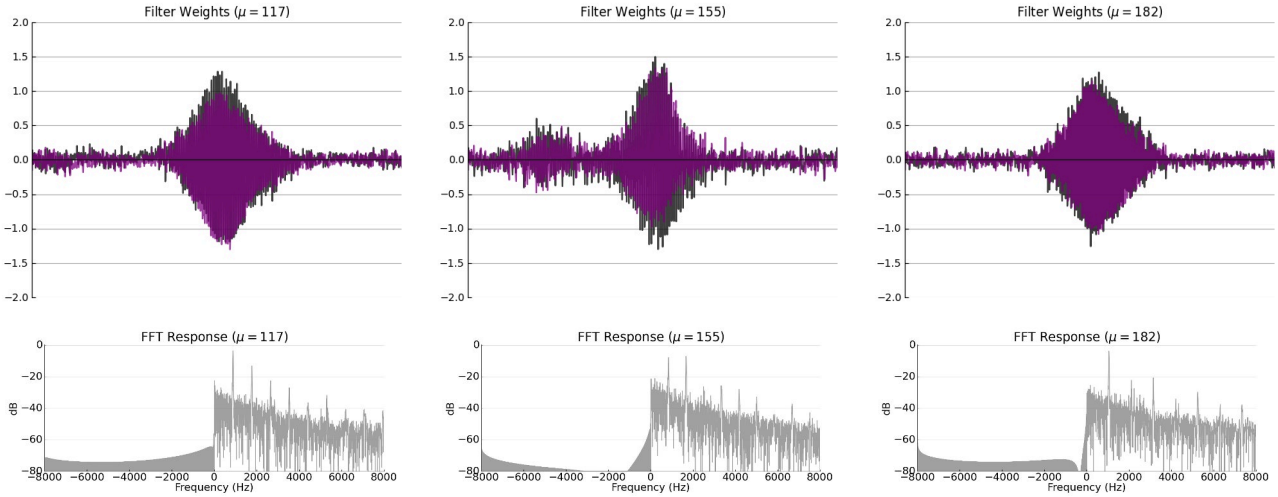


Figure 6. Examples of filters from the *hb+rnd+var* experiment which exhibit harmonic structure.

to more interpretable shapes in time and frequency. Indeed, it appears that the relevance of a filter can be directly estimated from its amplitude, with less important or even unused filters consisting of mostly near-zero weights rather than random noise. Furthermore, all of the Hilbert variant filters remain approximately analytic throughout the training process. We do not view the artifacts in the negative frequency range to be problematic, and believe they had a negligible effect on the outcome of experiments.

Some additional interesting qualities of the filters are observed, which are presumably related to the modeling of domain-specific characteristics for piano transcription. The spectrum of the filters in Fig. 4 include multiple prominent fundamental frequencies in clusters, which suggests the filters may be modeling harmony or polyphonic piano sounds. The filters in Fig. 5 are nearly symmetric, with two lobes in the impulse response that have overlapping or very close frequency peaks. These filters may be modeling the attack and decay of a piano note with the same pitch. Some of these filters, in addition to the filters in Fig. 6, exhibit harmonic patterns in the frequency spectrum. In particular, odd harmonics tend to be emphasized in these filters. These observations suggest that the proposed filterbank learning approach is capable of capturing lower-level features which are relevant for music transcription, and that the approach is likely extensible to similar tasks where this type of information is valuable.

The reduced performance of the filterbank learning models can potentially be attributed to too little training or inadequate tuning of hyperparameters. We also speculate that the consistency of piano data, coupled with large model complexity, may have led to overfitting, as evidenced by the lack of generalization to the MAPS dataset. State-of-the-art methods with no filterbank learning already perform piano transcription with reasonable proficiency. As such, it may be that a similar filterbank learning approach would show much more promise in areas with less data and less consistency, such as drum or guitar transcription.

Finally, we stress that the filterbank learning method described in this paper is not presented as a competitive alternative to state-of-the-art piano transcription methodologies. We do not offer a comprehensive comparison to such approaches for this reason, but do acknowledge that there are many better methods for the purpose of piano transcription. One may question the utility or validity of the filterbank learning methods due to their reduced performance. While superior performance would have been a nice outcome, we are still quite impressed with the performance of the filterbank learning models, especially the ones initialized randomly. From this perspective, it is clear the proposed methods, *i.e.*, filterbank learning, are successful for piano transcription, albeit not at the same level of performance as standard TFRs like CQT or Mel-spectrogram.

6. CONCLUSION

A filterbank learning module and various techniques were analyzed for the task of piano transcription. There are several reasons to suggest that the learned filters model note characteristics such as attack and decay, harmonic structure and relative strength, or inharmonicity. Given these are just hypotheses, a more objective explanation and analysis of the characteristics of the filters learned with our methods is left to future work. We showed it is possible to learn sparse filters from scratch, and that resulting filters differ significantly from those of the fixed transforms. Although the learned filterbanks did not surpass the performance of the fixed transforms, our method shows promise in learning to model time-domain note characteristics, and demonstrates that randomly initialized filterbanks can perform comparably to standard TFR initializations.

Acknowledgments

This work has been funded by the National Science Foundation grants IIS-1846184 and DGE-1922591. We would also like to thank Dr. Juan Cockburn and Dr. Andres Kwasinski for their guidance during preliminary work [4].

7. REFERENCES

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[2] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proceedings of ICLR*, 2019.

[3] A. Cogliati, Z. Duan, and B. Wohlberg, "Piano music transcription with fast convolutional sparse coding," in *IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2015.

[4] F. Cwitkowitz, "End-to-end music transcription using fine-tuned variable-Q filterbanks," Master's thesis, Rochester Institute of Technology, 2019.

[5] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proceedings of ISMIR*, 2018.

[6] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform CLDNNs," in *Proceedings of Interspeech*, 2015.

[7] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *Proceedings of ICASSP*, 2015.

[8] Y. Luo and N. Mesgarani, "TasNet: Time-domain audio separation network for real-time, single-channel speech separation," in *Proceedings of ICASSP*, 2018.

[9] ——, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 27, no. 8, pp. 1256–1266, 2019.

[10] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Proceedings of ICASSP*, 2014.

[11] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of ISMIR*, 2018.

[12] T. Kim, J. Lee, and J. Nam, "Sample-level CNN architectures for music auto-tagging using raw waveforms," in *Proceedings of ICASSP*, 2018.

[13] J. Lee, J. Park, K. L. Kim, and J. Nam, "SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, no. 1, 2018.

[14] J. Thickstun, Z. Harchaoui, and S. Kakade, "Learning features of music from scratch," in *Proceedings of ICLR*, 2017.

[15] R. G. C. Carvalho and P. Smaragdis, "Towards end-to-end polyphonic music transcription: Transforming music audio directly to a score," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017.

[16] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, "Learning filterbanks from raw speech for phone recognition," in *Proceedings of ICASSP*, 2018.

[17] N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, and E. Dupoux, "End-to-end speech recognition from the raw waveform," in *Proceedings of Interspeech*, 2018.

[18] S. Lattner, M. Dörfler, and A. Arzt, "Learning complex basis functions for invariant representations of audio," in *Proceedings of ISMIR*, 2019.

[19] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with SincNet," in *IEEE Spoken Language Technology Workshop (SLT)*, 2018.

[20] N. Zeghidour, O. Teboul, F. d. C. Quitry, and M. Tagliasacchi, "LEAF: A learnable frontend for audio classification," in *Proceedings of ICLR*, 2021.

[21] M. Won, S. Chun, O. Nieto, and X. Serrc, "Data-driven harmonic filters for audio representation learning," in *Proceedings of ICASSP*, 2020.

[22] Z. Zhang, Y. Wang, C. Gan, J. Wu, J. B. Tenenbaum, A. Torralba, and W. T. Freeman, "Deep audio priors emerge from harmonic convolutional networks," in *Proceedings of ICLR*, 2020.

[23] E. Vincent, N. Bertin, and R. Badeau, "Adaptive harmonic spectral decomposition for multiple pitch estimation," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 18, no. 3, pp. 528–537, 2010.

[24] T. Cheng, M. Mauch, E. Benetos, S. Dixon *et al.*, "An attack/decay model for piano transcription," in *Proceedings of ISMIR*, 2016.

[25] A. Cogliati, Z. Duan, and B. Wohlberg, "Piano transcription with convolutional sparse lateral inhibition," *IEEE Signal Processing Letters*, vol. 24, no. 4, pp. 392–396, 2017.

[26] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 24, no. 5, pp. 927–939, 2016.

[27] J. W. Kim and J. P. Bello, "Adversarial learning for improved onsets and frames music transcription," in *Proceedings of ISMIR*, 2019.

[28] R. Kelz, S. Böck, and G. Widmer, "Deep polyphonic ADSR piano note transcription," in *Proceedings of ICASSP*, 2019.

[29] T. Kwon, D. Jeong, and J. Nam, "Polyphonic piano transcription using autoregressive multi-state note model," in *Proceedings of ISMIR*, 2020.

[30] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution piano transcription with pedals by regressing onset and offset times," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 29, pp. 3707–3717, 2021.

[31] Y. Yan, F. Cwitkowitz, and Z. Duan, "Skipping the frame-level: Event-based piano transcription with neural semi-CRFs," in *Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2021.

[32] M. Pariente, S. Cornell, A. Deleforge, and E. Vincent, "Filterbank design for end-to-end speech separation," in *Proceedings of ICASSP*, 2020.

[33] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proceedings of ICML*, 2017.

[34] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, "Sequence-to-sequence piano transcription with transformers," in *Proceedings of ISMIR*, 2021.

[35] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 18, no. 6, pp. 1643–1654, 2010.

[36] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common MIR metrics," in *Proceedings of ISMIR*, 2014.

[37] C. Schörkhuber, A. Klapuri, N. Holighaus, and M. Dörfler, "A matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution," in *Audio Engineering Society (AES) 53rd Conference on Semantic Audio*, 2014.

REDESIGNING A PIANO ROLL: A MELODY INPUT INTERFACE THAT CAN PLAY MICROTONES WITH AN ARBITRARY NUMBER OF KEYS

Tatsunori Hirai
Komazawa University
thirai@komazawa-u.ac.jp

ABSTRACT

A piano roll plays a fundamental role in computer music production. As its name suggests, a piano roll is designed with the motif of a piano keyboard. In DAW software, synthesizers can produce a variety of synthesized sounds and simulated instrument sounds. Although various synthesized sounds and simulated instrumental sounds can be produced using synthesizers, the most common input format is a piano roll with keyboard input.

Although keyboard input is simple, especially for those familiar with it, it has a significant limitation in that notes that can be played are limited to those in 12 equal temperament. It is possible to change the actual frequency of keyboard sound by changing the tuning settings (which is called calibration). However, an input interface based on the concept of a piano keyboard has many limitations in terms of inputting microtones in cases where an octave is not composed of 12 tones (e.g., 17 and 24 equal temperament). Based on these problems, we propose an interface that allows users to customize temperament that can be played by a piano roll and to redesign a piano roll. Specifically, our interface allows users to move back and forth between editing temperaments and creating melodies. The temperament can be created in three different ways. Creating from scratch, editing an existing template temperament, or creating from an imported external audio file.

1. INTRODUCTION

The piano roll is an indispensable interface for computer music production. In DAW software for computer music production, it is common to input notes using a piano roll interface, even for instruments that do not use a keyboard, such as strings and wind instruments. A piano keyboard consists of 12 keys per octave, divided into seven white keys and five black keys. This keyboard interface is simple for those familiar with music, but has significant limitations: notes that can be played are limited to those of 12 equal temperament.

12 equal temperament is a quantized temperament that uses only 12 patterns of frequencies in an octave. A piano roll is used to input the sounds of instruments with a playing style significantly different from that of the piano. For stringed instruments without frets, such as the violin, the number of notes that can be played in an octave is not limited to 12, but when using piano rolls, the number is limited to 12. The violin may be played in a different temperament other than 12 equal temperament, for example just intonation; however, 12 equal temperament is generally used with piano roll. Most DAW software has a tuning function that allows a user to change the temperament that can be played on the piano roll. However, as long as the keyboard interface is used, the constraints of the number 12 cannot be avoided. To achieve more flexible music production, an interface that removes the restrictions of the piano roll and allows the input of notes of any tone, not limited to 12, including microtones, is required.

In this paper, we propose an alternative input interface to the piano roll, where a user can design the number of keys per octave and their respective frequencies. The proposed note input interface allows users to design their original musical temperament, avoiding the constraints of existing musical temperaments and piano roll keys.

Some musical equipment, such as samplers, enables music production not constrained by musical temperament, but it is difficult to compose melodies and accompaniments using samplers. The proposed interface also determines temperaments based on the sampling of external sound data. For example, the proposed interface enables the design of keys and temperaments from the sounds of birds, waves, and wind.

Copyright: © 2022 Tatsunori Hirai et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. RELATED WORK

Research related to microtones is called n-EDO in the context of n-equal temperament other than 12 equal temperament and has been studied by microtonal researchers such as Erv Wilson. EDO stands for equal divisions of an octave. Erv Wilson found that the sounds of 17, 19, 22, and 31-EDO were comfortable and explored them [1]. Music theory research on new temperaments is important for discovering new temperaments; however, music can be made also without a formal knowledge of music theory just like music when music theory did not exist. In this study, we create tools prior to music theory.

Research on microtonal instruments has been conducted to explore tools that can play microtones. Bailey et al. [2] proposed microtonal clarinet, and Dabin et al. [3] proposed microtonal flutes by 3D modeling and printing. These instruments enable us to play music that differs from existing musical temperaments; however, physical instruments are not flexible and cannot be easily changed to other tempera-

ments. If a performer wants to play a piece of music in various temperaments, the performer will need to design and use a different instrument for each temperament. For example, various types of guitars have been proposed to play microtones [4]; however, it is difficult to realize flexible temperament because of a fixed parts such as frets. Therefore, guitars that use frets have to be designed for each temperament. There are also fretless guitars, but the degree of freedom is directly related to the difficulty in playing, so it is not an instrument that everyone can easily handle.

Research on tuning methods [5] and perception of microtones [6] has been conducted from a variety of perspectives [7], which has contributed to the development of microtonal music. On the other hand, we believe that developing a tool in advance and allowing users to freely design their temperaments will lead to the development of microtonal music through an example-based approach that differs from an approach of conventional music theory research.

The generalized keyboard proposed by Bosanquet in the 1870s is a keyboard capable of playing microtones, and was later extended by Erv Wilson and other microtonal researchers [8]. It is possible to play microtonal music with the generalized keyboard, but since its structure is significantly different from the piano keyboard we are familiar with, it is difficult to introduce it directly into the existing music production flow. In addition, there is a disadvantage in that it is difficult to intuitively understand notes that correspond to each key when considering frequent temperament changes. There is also a software for tuning microtonal scale called Scala [9]. Scala is a software that enables easy tuning of any temperament; however, it does not provide a function for music production like a piano roll.

Some existing music production software allows users to tune to any frequency when inputting notes to a piano roll. Software synthesizers such as PianoTeq [1] and Serum [2] are examples of such software. It is also possible to import and use tuning files related to a user's original temperament designed on sites such as Scale Workshop [3]. However, existing software synthesizers are designed to be programmed with piano rolls consisting of 12 keys per octave. Leimma [4] provides browser-based software to create temperament consisting of any number of notes with circular representation of the octave; however the melody creation part using the created temperament has not been considered and the number of keys on the keyboard is fixed to 12.

In our proposed interface, we enable users to redesign the input notes for piano rolls, while referring to the tuning methods of existing software such as PianoTeq and Leimma. These tuning methods employ a circular representation of an octave, which is not so new, as it can also be seen in other research paper, for example in [10]. What makes our interface particularly different from other software is that a user can go back and forth between melody input using the piano roll-like interface and temperament creation. In addition, the proposed interface provides a function to generate temperament from imported external audio files.

[1] https://www.modartt.com/pianoteq
[2] https://xferrecords.com/products/serum
[3] https://sevish.com/scaleworkshop/
[4] https://isartum.net/leimma

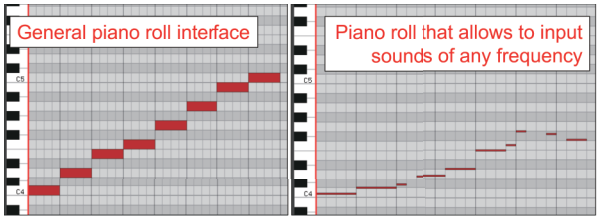


Figure 1. Implementation of a piano roll that allows input of sounds of arbitrary frequency.

3. A STRATEGY TO INPUT FREE TEMPERAMENT MUSIC

In this study, we investigate how to extend the existing interface of a piano roll to enable the input of notes with microtones and users' original temperaments. As an initial study, we implemented an interface in which quantization in the vertical axis direction (i.e., frequency direction) was disabled in the piano roll.

A comparison between the general piano roll and the piano roll implemented here, which allows the input of notes of any frequency, is shown in Figure 1. This interface allows the input of any arbitrary frequency microtonal notes. At first glance, this may seem to achieve the goal, but with a high degree of freedom in the frequency direction, it is difficult to maintain musical discipline and create melodies that have a pattern in pitch. MetaSynth [5], an existing music production software also has an input interface not quantized in the frequency direction. Although it is possible to produce pitches outside the equal temperament with such an interface, it is difficult to create a series of sounds that can be considered as music by many listeners. The introduction of microtones expands the range of music that can be produced. However, music production using microtones is difficult because microtonality does not have a widespread fully standardized set of compositional rules. The greater the freedom of the temperament, the more difficult it is expected to be to compose music.

While maintaining the goal of increasing the degree of freedom of programmable sounds, constraints are also necessary to make the music work as music. In general piano rolls, the constraint of 12 equal temperament and the music theory based on it guarantee the musical structure. Based on the above, the proposed interface does not provide complete freedom when programming notes, but provides freedom in key design. Note input can be performed in the same procedure as with conventional piano rolls, but the constraints of the number of keys (12 keys per octave) and the frequency assigned to each key existing in piano rolls are removed in our proposed interface.

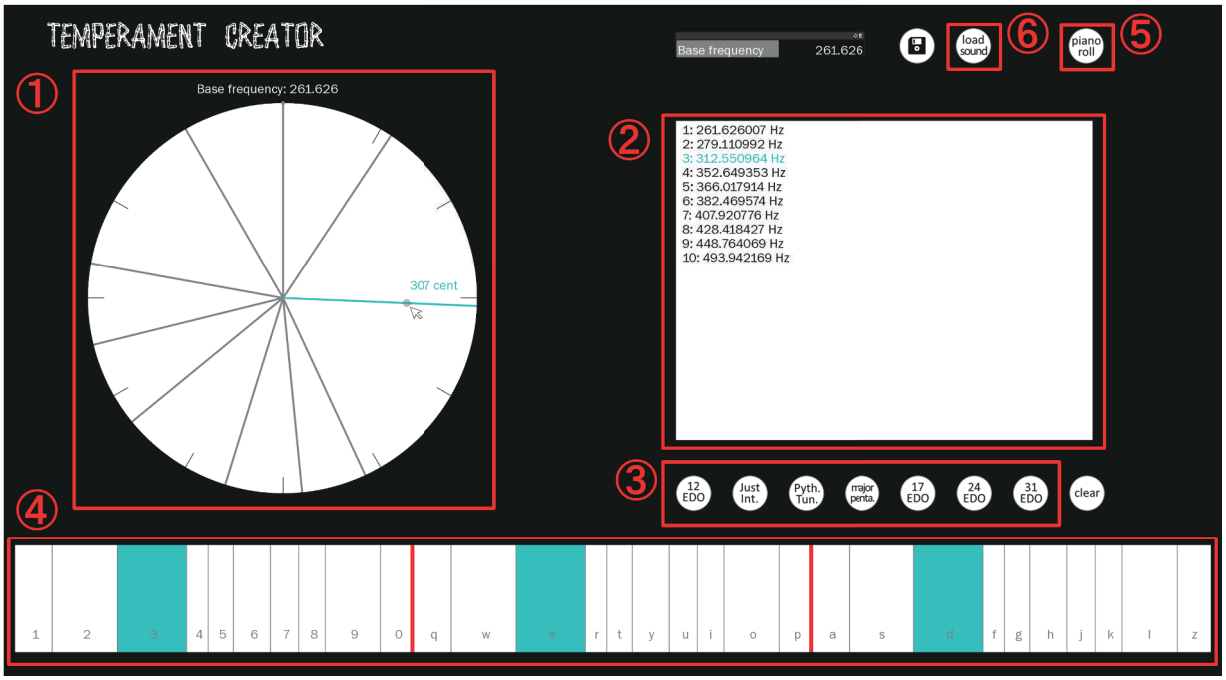


Figure 2. Screen capture of the tuning function of the proposed system.

4. SYSTEM OVERVIEW

In this section, we describe the overall structure of the proposed system. The proposed system consists of two interfaces: one for creating a custom temperament, and the other for programming music based on the created temperament. Figure 2 illustrates the screen capture of the temperament creator interface.

The pie chart on the left (1) is an interface that enables users to adjust the frequency of each key in cents [6], and to add or delete new keys, with an octave being one cycle. The lines on the circle represent each key. The frequency corresponding to each keyboard key is displayed in a text format on the right side (2). This pie chart interface is implemented with reference to PianoTeq's advanced tuning mode. The main difference between our interface and PianoTeq is that our interface allows users to create temperaments from scratch, and the number and the width of keys can be determined arbitrarily.

4.1 Loading Temperament Information from Templates

The proposed interface is implemented as an extension of a piano roll, so it can be used to input notes in the same way as a piano roll. It is possible to create a keyboard with 12 equal temperament by selecting one of the available temperament templates (3). It is also possible to create other keyboards, such as just intonation, Pythagorean tuning, major pentatonic scale, 17-EDO, 24-EDO, and 31-EDO with only one click of a button. Although these temperaments and scales do not necessarily have 12 notes per octave, the interface is designed to ensure that they can all be handled within the same scheme.

When creating a musical temperament, the corresponding key is reflected in the lower part of the interface (4) so that a user can play a note to check the sound. The keyboard is displayed for three octaves. The width of each key corresponds to the interval. In the case of equal temperament, the keys are spaced uniformly, but in the case of a temperament in which the intervals are not uniform, the spacing of keys varies according to the intervals. This is equivalent to the function of flexibly moving the position of frets in a stringed instrument with frets, such as a guitar. The interval steps of common keyboard and string instruments are uniform in semitones, but they are not always uniform in microtones. By changing the width of the keyboard according to the interval, a user can intuitively understand the interval between the left and right keys. The user can easily imagine the pitch of the sound just like fretless guitar. Our proposed keyboard interface follows the principle that the leftmost position produces lower tones and the rightmost position produces higher tones, making it more intuitive than the interface of generalized keyboard.

4.2 Piano-roll-like Melody Input Interface

Once the temperament is decided, a user can press the piano roll button in the upper right corner (5) to switch to the melody input interface. The switched interface screen is shown in Figure 3.

The interface in Figure 3 is almost the same input interface as a typical piano roll. The difference is that sounds are based on the user's original temperament, and there is no distinction between white and black keys since the interface is no longer based on the piano. The only reason

[5] https://uisoftware.com/metasynth/

[6] A cent is a unit of 1,200 equal divisions of an octave. In the case of the 12 equal temperament, a semitone is 100 cents.

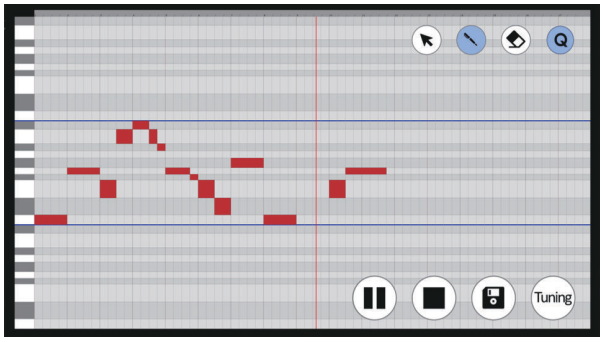


Figure 3. Our piano-roll-like melody input interface. Each note has a different height based on a created temperament.

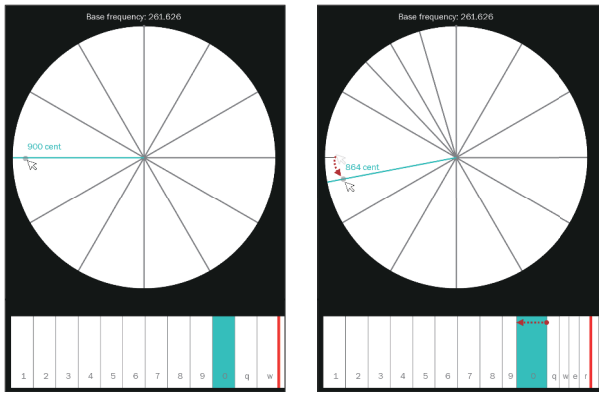


Figure 4. Editing tones of a temperament template to customize notes in a scale. An example of changing the tuning from 12 equal temperament and adding some tones. The width of keys on the keyboard changes flexibly according to the interval.

why our proposed interface has bright and dark keys is to make it easier to distinguish them from adjacent keys.

In the current implementation, only two types of tones are available when a melody is played back with this interface: sinusoidal wave and rectangular wave tones. Although timbre is an important factor in creating melody, our interface currently does not take it into account, and we are considering to extend the interface to enable it to work with external sound sources in the future.

4.3 Customization of Temperament

By clicking on the tuning button at the lower right of Figure 3, a user can return to the temperament creation interface and tune the temperament while editing the melody.

Figure 4 shows the editing of 12 equal temperament keys to create original temperament keys. As shown in Figure 4, by editing the lines on the pie chart, a user can change the frequency of notes assigned to each key, as well as add and delete keys.

In general, tuning of musical instruments is performed by setting the frequency of the note A4 to 440 Hz or 442 Hz and using that as the basis. However, since the proposed interface is not limited to 12 types of pitch classes, it is inappropriate to use A4 as the basis. Therefore, the proposed

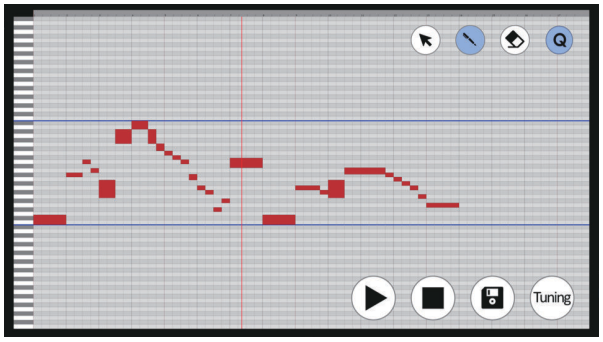


Figure 5. Mixing multiple temperaments in one melody.

interface calculates the frequencies of other keys based on the frequency of the lowest note in the octave (e.g., note C in the case of 12 equal temperament).

4.4 Mixing Temperaments in One Melody

Users can switch between the temperament creation interface and the piano roll-like melody input interface as many times as they want. In addition, one of the unique features of the proposed interface is its ability to mix multiple temperaments. It is possible, for example, to enter a melody in 12 equal temperament for the first half of the melody and then in 17-EDO for the latter part. Modulation is a common concept in music, but with our interface, it is possible to change temperaments during a tune.

Figure 5 shows a melody with 24-EDO temperament added to the melody that was programmed with the original temperament shown in Figure 3. In the proposed interface, it is possible to change the temperament flexibly while editing a melody, so that a user can change any single note to a note that is not included in the original temperament. There are times when a composer may want to intentionally put a note outside the current temperament while composing, but in the case of 12 equal temperament, the option of a wrong note is limited to few options. Since the proposed method can assign a note of any frequency to a key, it provides a variety of options for a wrong note.

4.5 Creating Temperaments from External Audio File

Although a temperament can be easily created using our pie chart interface, it is necessary to go through a trial and error process to find a temperament that can be considered music. It would not be difficult to create a temperament with minimum musical problems by editing a temperament template, but the originality would be diminished if an existing temperament was used as a basis. To solve this problem, we implemented a function to create temperaments from an existing audio file. A user can import external audio file to be used to create a new temperament by clicking on the "load sound" button at the upper right (6) of Figure 2.

The expression method of sampling, which uses sounds of everyday life to create music, has become popular. Our interface enables sampling for temperament creation such that frequencies used in everyday sounds are assigned to

#	Temperament	Frequencies [Hz]
1	12 equal temperament	[261.63, 277.18, 293.67, 311.13, 329.63, 349.23, 369.99, 392.00, 415.31, 440.00, 466.16, 493.88]
2	C-major pentatonic scale	[261.63, 293.67, 329.63, 392.00, 440.00]
3	Just intonation	[261.63, 279,07, 294.33, 313.96, 327.04, 348.84, 370.64, 392.45, 418.60, 436.05, 470.93, 490.56]
4	24-EDO	[261.63, 269.29, 277.18, 285.31, 293.66, 302.27, 311.13, 320.24, 329.63, 339.28, 349.23, 359.45, 369.99, 380.84, 392.00, 403.48, 415.31, 427.47, 440.00, 452.89, 466.16, 479.82, 493.88, 508.36]
5	Mixture of 12 equal temperament and Just intonation	[261.63, 277.18, 279,07, 293.67, 294.33, 311.13, 313.96, 327.04, 329.63, 348.84, 349.23, 369.99, 370.64, 392.00, 392.45, 415.31, 418.60, 436.05, 440.00, 466.16, 470.93, 490.56, 493.88]
6	Randomly created 5 tones	[261.63. 290.81, 362.37, 419.01, 473.63]
7	Randomly created 12 tones	[288.06, 310.53, 323.34, 338.66, 362.94, 392.49, 406.08, 426.86, 464.17, 497.83, 517.09, 550.42]
8	7 tones sampled from from chicken crows	[341.30, 423.10, 506.52, 540.17, 580.01, 629.65, 681.66]
9	7 tones sampled from the sound of ripples	[323.00, 343.19, 365.71, 416.46, 463.43, 532.34, 603.61]

Table 1. Examples of created temperaments.

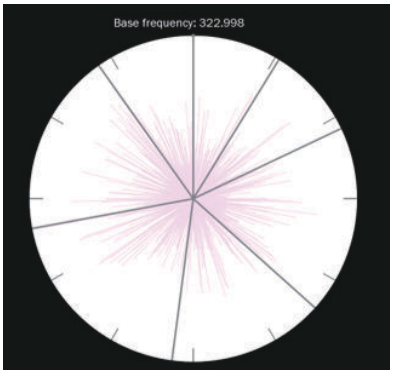


Figure 6. Creating a temperament from external audio file. The light red area in the pie chart corresponds to the spectrum of the imported audio data.

keys. For example, by importing an audio file of bird sounds, users can design a keyboard with the frequencies of bird sounds.

We are unbothered if the birds do not sing at the frequency of 12 equal temperament. From this, we thought that the frequencies of birdsong and ocean waves could be sublimated into music. Whether the actual temperament created from the external audio files will be musically good depends on how it is used, but we believe that it is a function that makes it easier to create a temperament than starting from scratch.

Figure 6 shows a pie chart example for the actual case of creating a temperament from the sound of a chicken clucking. The spectrum of the audio file is represented by the line in light red on the pie chart. Here, the most predominant spectral components for a time interval of audio length are selected as tones. This is accomplished by extracting the peak of the spectrum within a predetermined frequency range and assigning keys to a predetermined number of peaks. In the example shown in Figure 6, a range of one octave is set by searching for peaks within the range of 200-600 Hz and using the highest peak as the reference frequency. Tone frequencies outside the octave range are adjusted using octave equivalence to keep them within the range. Here seven peaks of the spectrum are searched and assigned to keys.

5. EXAMPLES OF CREATED TEMERAMENTS

Table 1 summarizes several examples of temperaments created with our proposed interface. The temperaments from no.1 to no.4 can be generated from the prepared templates at the click of a template selection button. The most basic way to use our interface is to load the prepared template template and tune them. This corresponds to the general tuning of the instrument. Alternatively, the template temperament can be used as is, in which case the proposed interface is equivalent to a conventional piano roll when a 12 equal temperament is used. Many softwares such as PianoTeq, Scale Workshop, and Leimma provide these templates as the basic function.

The temperament no.5 is the mixture of 12 equal temperament and just intonation. Since the frequencies of the lowest tones are the same in our interface, there are 23 tones in total [7]. With this temperament, the user can choose whether to use the 12 equal temperament or the just intonation for every single note. The capability to realize a piano roll with a mixture of multiple temperaments is a unique feature of the proposed interface.

[7] Generally, the frequency of the note A4 is set to 440 Hz in just intonation. The proposed interface creates a temperament based on the frequency of the lowest note.

The temperaments of no.6 and no.7 are randomly created temperaments using our proposed interface. It is up to the user to create a listenable melody from these temperaments. However, with this interface, it is possible to create a melody in the piano roll while changing the temperament, and vice versa. Through repeated trial and error, both originality and uniqueness of the melody can be pursued. The number of keys in an octave is expected to increase with repeated trial and error, however, there is no limit to the number of keys per octave in our interface as long as it can be displayed on the screen. Although Leimma has a similar temperament creation function, Leimma only provides the temperament creation part and does not support the creation of melodies or editing of temperament while creating melodies.

The temperaments of no.8 and no.9 are temperaments created from imported external audio files. The temperament no.8 is a temperament created based on the peak information in the spectrum of a 2.5-second-long chicken crows. The temperament no.9 is a temperament sampled from the sound of ripples which is 1.5-second-long. Here, seven peaks are extracted for each temperament, however, since the spectrum itself is visualized, it is easy to increase the number of keys to be assigned according to the user's preferences.

The several temperaments shown here are only examples of the temperaments that can be freely created with the proposed interface. In the same way that the user can freely create melodies in the piano roll, the temperament can be freely edited with this interface.

The evaluation of the usability of the proposed interface is a topic to be addressed in the future research because it depends on the subject's skill of using DAW software. Whether the melody is good or bad all depends on the user's ingenuity. We will investigate how to support composing with the newly defined temperament in our future study.

6. DISCUSSION

In this study, we proposed an interface that allows users to freely create original temperaments and redesign piano rolls. For a long time, the concept of musical temperament had been static, and users were rarely allowed to change it freely, only to modify the tuning of existing instruments. Conventionally, notes in an octave were divided into 12 tones, and it was difficult to express a finer interval change.

12 equal temperament is the most popular musical temperament of all time. Nonetheless, other temperaments have also been used, especially in Eastern Asia and Arabic music. In addition, in contemporary music, temperaments other than 12 equal temperament, such as microtones, are essential. These examples demonstrate that there is a good chance that other attractive temperaments exists that we have not yet encountered. We believe that exploring musical temperaments will lead to the discovery of new music.

According to the sound dissonance curve proposed by Helmholtz [11], 12 equal temperament tones are not composed of only the notes with low dissonance. If we consider only dissonance, we can say that just intonation is the best temperament. The reason for the popularity of 12 equal temperament is that they are easy to transpose keys. In this sense, musical temperament may be something that can be more freely defined.

Because the temperaments of a musical instrument influence its design, changing the temperament is difficult if the same physical instrument is to be used. However, in this age, music production using computers has become common, and flexibly changing musical temperaments using software is possible. In the future, when everyone is free to design their original musical temperament, we may be able to discover rules for how to create good musical temperaments and how to identify bad ones.

Tones in temperaments other than 12 equal temperament are often unfamiliar. This could simply be because our ears are not accustomed to them. As the number of music with such temperaments increases, it can be expected that our perception for sounds other than 12 equal temperament will be developed. To achieve this, it is necessary to increase the amount of such music data in the future. We believe that the proposed interface will help increase the number of music with such new temperaments.

New temperaments are often not applicable to well-known musical theories. Thus, it is necessary to develop technology to assist in composing music in temperaments encountered for the first time, and this will be our future task. In addition, we would like to build an environment, where anyone can use the proposed system by converting it into a VST plug-in. Although only sinusoidal wave and rectangular wave sound can be played on the current interface, we are considering to realize melodies with richer tones using existing sound sources by converting to VST plug-in. Since the impression of a melody changes depending on the tone of the melody, the exploration of the tone is also a future research topic.

Currently, the proposed interface does not take into account the components of music other than the melody. In making music, we must also consider harmony, accompaniment, and many other elements. In the future, we would like to extend our interface to remove the restriction of existing temperament in various musical elements.

The evaluation of the proposed interface is a major issue to be addressed in the future. We are considering to conduct a user study in which subjects actually operate our interface. To do so, we need to carefully consider the evaluation method, since it is expected to be greatly affected by how the subjects are accustomed to the conventional temperaments, and most listeners are 12 equal temperament educated listeners. If a user study were to investigate whether the flexibility of the melodies that can be created has been increased, it would be obvious without the need to conduct the study. The important points in the evaluation are how much the increased flexibility of the melody contributes to the creation, the usability of the interface, the ease of composing, and the adequecy of the interface design. We are currently considering a method to realize a user study that removes various influences as much as possible by carefully selecting subjects to be evaluated.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number JP19K20301. We thank the reviewers for their valuable and insightful comments.

7. REFERENCES

[1] T. Narushima, *Microtonality and the tuning systems of Erv Wilson*. Routledge, 2017.

[2] N. J. Bailey, T. Cremel, and A. South, "Using acoustic modelling to design and print a microtonal clarinet," in *Proceedings of the 9th Conference on Interdisciplinary Musicology (CIM14), Berlin, Germany*, 2014, pp. 4–6.

[3] M. Dabin, T. Narushima, S. T. Beirne, C. H. Ritz, and K. Grady, "3d modelling and printing of microtonal flutes," 2016.

[4] J. Schneider, "The microtonal guitars of harry partch," *Soundboard Scholar*, vol. 1, no. 1, p. 5, 2015.

[5] W. A. Sethares, "Adaptive tunings for musical scales," *The Journal of the Acoustical Society of America*, vol. 96, no. 1, pp. 10–18, 1994.

[6] F. Bailes, R. T. Dean, and M. C. Broughton, "How different are our perceptions of equal-tempered and microtonal intervals? a behavioural and eeg survey," *PloS one*, vol. 10, no. 8, p. e0135082, 2015.

[7] G. J. Balzano, "The group-theoretic description of 12-fold and microtonal pitch systems," *Computer Music Journal*, vol. 4, no. 4, pp. 66–84, 1980. [Online]. Available: http://www.jstor.org/stable/3679467

[8] D. Keislar, "History and principles of microtonal keyboards," *Computer Music Journal*, vol. 11, no. 1, pp. 18–28, 1987. [Online]. Available: http://www.jstor.org/stable/3680175

[9] Manuel Op De Coul, "Scala." [Online]. Available: https://www.huygens-fokker.org/scala/

[10] A. Barate and L. A. Ludovico, "Generalizing messiaen's modes of limited transposition to a n-tone equal temperament," in *Sound and Music Computing*, 2015, pp. 287–293.

[11] H. L. Helmholtz, *On the Sensations of Tone*. Dover, New York, 1954.

A COMPARISON OF PITCH CHROMA EXTRACTION ALGORITHMS

Miguel Perez[♯♭], Holger Kirchhoff[♯], Xavier Serra[♭]
Huawei Technologies Munich Research Center[♯]
Music Technology Group, Universitat Pompeu Fabra[♭]
miguel.perez.fernandez@huawei.com
holger.kirchhoff@huawei.com
xavier.serra@upf.edu

ABSTRACT

The pitch chroma is a popular way to represent pitch information in an octave independent way, with applications in automatic chord recognition, cover song identification, audio-to-score alignment, and others. Early chroma extraction algorithms employed expert knowledge to derive pitch chromas from short-time spectra. With the rise of deep learning, the emphasis moved from algorithm design to the structure of the network and the selection of appropriate training data. The approaches perform differently for various types of audio input. We conducted a set of experiments in order to explore the qualitative properties that each algorithm exhibits. These include how the number of concurrent pitches influences the chroma representation, and how noise or unpitched percussion can degrade the performance of the algorithms. We performed a quantitative analysis of various algorithms under these scenarios. The results show that chromas based on deep learning show huge potential, especially when it comes to noise reduction and ignoring non-tonal aspects of the music. However, we also found that some deep learning based chromas fail to accurately detect pitches at lower polyphony levels. We reflect on these results and discuss some paths to improvements for future chroma extraction algorithms.

1. INTRODUCTION

Data representation is an important aspect in designing systems that analyze and process real-world data. In the case of natural language processing it is common to use the words themselves with special tokens to indicate the beginning and end of phrases. In computer vision, the RGB channels of the images are used to feed the algorithms. In both of these areas, an unprocessed representation of the data can be used as the input for the systems. For music data, approaches that directly act on time-domain sample data have been less common. This can be attributed to the fact that the waveform representation is inherently difficult to interpret for humans and hence makes it difficult to design expert systems without any intermediate representation. More practically, machine learning systems that act on waveform data turned out to be significantly harder to train and usually result in larger model sizes that require more training data [1]. Intermediate representations that transform the raw audio into some more interpretable representation, however, have proven useful even for machine learning systems, as they can reduce the dimensionality of the input data and often lead to more accurate and robust results. Since music can be understood in different aspects, such as rhythm, melody, instrumentation, genre, etc., different intermediate representations such as STFT spectrograms, CQT spectrograms or Mel spectrograms, or features derived from these have been extensively used for analysis algorithms.

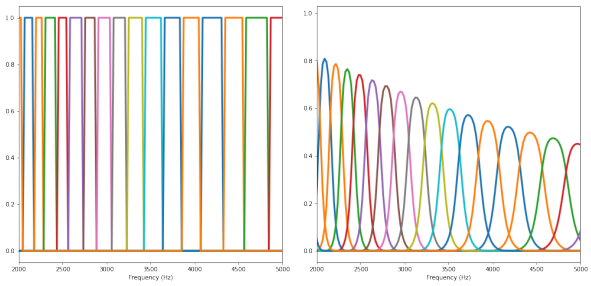
The pitch chroma is a feature addressing the tonal information contained in a music signal. It is sometimes denoted as pitch-class profile (PCP) or simply as chroma. It encodes tonal information of music in an octave independent representation, also known as pitch-class. A chroma is a vector of usually 12 dimensions, each representing the presence of a pitch class (C, C#, D, . . . , B). Finer pitch resolutions like quarter tones (a chroma with 24 dimensions) can be used, but are less common in the literature. Chromas are usually extracted for short, consecutive blocks of audio, resulting in a *chromagram* representation for an extended section of music. Pitch chroma representations can be considered more robust than those accounting for octave height, since the compression to a single octave eliminates wrong pitch estimates in a different octave (octave errors) caused by ambiguous harmonic patterns in the spectrum. Although initially designed for automatic chord estimation (ACE), chromas became useful in a wide range of tasks, such as cover version identification (CVI) [2], audio-to-score alignment [3] and music creation [4].

Given the above definition and applications for pitch chromas, a chroma extraction algorithm is expected to fulfil a number of tasks when transforming audio to chromas. Generally, a chroma extractor should eliminate any type of irrelevant spectral content in order not to obscure the tonal information. Irrelevant content can be any transient, non-tonal components (percussive or noise-like) such as drums, cheering audience, etc. We here neglect the fact that percussive instruments might also have a tonal component to them. In the same way, overtones should not contribute to pitch classes other than their corresponding fundamental frequency. It is arguable whether a chroma should represent the tonal content *as it is*, or whether it should incorporate further processing steps depending on the target

Copyright: © 2022 Miguel Perez et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

application, e.g temporal smoothing in the case of ACE. In any case, a transparent processing structure would require a chroma to only represent those pitch classes that are present at each point in time. Subsequent processing stages would decide then, if a set of notes belong to an arpeggiated chord, or if some notes are ornaments that do not belong to the chord itself.

The rest of the paper is divided into four sections. In Section 2 we give an overview of the most popular algorithms, highlighting the main differences and evolution over time. In Section 3 we describe the experiments we have carried out in order to do a qualitative analysis of the different approaches. The results derived from the different experiments are presented in Section 4. Section 5 contains our reflections on the properties of the different chromas and possible future research directions.



(a) Fujishima's [5] filters (b) Ellis & Poliner filterbank [6]

Figure 1: Filterbanks in the 2-5 kHz range. Each color represents a different pitch class. Fujishima rectangular filters make abrupt changes between pitch assignments. Ellis & Poliner penalize the deviations from the ideal frequencies.

2. RELATED WORK

In this section, we present an overview of various notable approaches to chroma extraction. We divided the algorithms into three different classes: knowledge-based chromas, deep chromas from chord labels, and multipitch deep chromas. In the first category, we included chroma algorithms that only use expert knowledge and digital signal processing techniques to obtain chroma representations. The second one contains chroma algorithms that make use of deep learning with ACE datasets. Lastly, we introduce chroma algorithms trained with datasets containing pitch annotations instead of chord labels.

2.1 Knowledge-based Chromas

Chroma algorithms evolved since the first algorithm presented by Fujishima [5]. In his work, he presents a bank of non-overlapping rectangular filters (see Figure 1a) that maps STFT magnitude spectra to pitch classes. After applying the filters to the spectrum, the energy of all filter outputs belonging to the same pitch class are accumulated to form the pitch chroma. The fact that the filters are non-overlapping makes them quite selective w.r.t the frequency components. The rectangular shape of the filters has the effect that all frequency components in a semitone range contribute equally to the pitch class. Since the tonal components of interest will usually appear closer the the center of the filter, it might be beneficial to give less weight to components that deviate from the center, in order to reduce the influence of non-tonal (noise, percussion) and spurious components. Following this idea, Ellis & Poliner [6] proposed a filterbank of Gaussian filters with a semitone spacing. With this shape, frequency components further away from the center of the filter will contribute less to that specific pitch class. Also, to reduce the influence of percussion or other elements, the filters emphasize mid-frequencies, where most of the tonal content will be located. See Figure 1 for a comparison of Fujishima and Ellis & Poliner filters.

With both these approaches, energies of harmonics which contribute to the pitch perception of their fundamental are here erroneously assigned to different pitch classes. E.g. the 3rd harmonic is a perfect fifth above the fundamental and will hence not contribute to the pitch class of its fundamental, but to the pitch class a fifth above. This leads to false contributions to the resulting pitch chroma. Gómez [7] therefore proposes to extract *harmonic* pitch-class profiles (HPCP). This method differs from the previous methods in two significant ways: Firstly, it only considers peaks of the spectrum instead of complete frequency bands, and it secondly also maps harmonics to the pitch class of their fundamental frequency. The use of peaks aims to reduce the influence of undesired elements, such as unpitched percussion or background noise. The mapping of harmonics is achieved by accumulating the energies not only of the fundamentals but also of their overtones with decreasing weight. The contribution of the harmonics to the fundamental can be seen as a pattern-matching mechanism.

The NNLS chroma proposed by Mauch & Dixon [8] employs another pattern matching mechanism to identify harmonic structures in the short-time spectrogram. Their system first maps STFT magnitude spectra onto a log-frequency axis with a 1/3-semitone resolution. Given a dictionary of prototypical spectra for a pitch range range from A0 to G#6, approximate note activations are extracted by means of a non-negative least squares (NNLS) algorithm. The note activations are then summarized to form the pitch classes for each instantaneous chroma.

Another popular chroma representation with pattern matching mechanisms is the ***c**hroma DCT-**r**educed log **p**itch* (CRP) [9]. This approach is inspired by mel-frequency cepstral coefficients (MFCCs), a popular representation for speech, which produces a set of coefficients where the firsts ones are closely related to timbre [10]. During the MFCC extraction process, the discrete cosine transform (DCT) captures periodicities present in the spectrum. After mapping the STFT magnitude spectra onto a log-frequency axis with a semitone resolution, the authors apply the DCT as it is done during the MFCC extraction process to extract a number of coefficients. The information concerning timbre is discarded by setting the first n coefficients to zero, and the inverse DCT is expected to return a chroma with improved robustness to timbre. Both the number of coefficients to

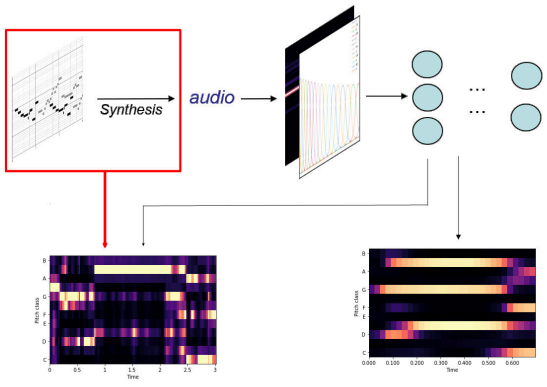


Figure 2: The general schema to train deep chromas. The audio signal is converted to an intermediate representation such (e.g: CQT), which is used as input for the neural network that returns the chromagrams. The red box illustrates the case of Wu & Li [11] where a MIDI score is used to create the audio and later to serve as the ground-truth

extract and to discard are parameters that must be set according to the use case.

2.2 Deep Chromas: Chord Labels

Pattern matching mechanisms help to distinguish between fundamentals and harmonics, but also add even more parameters in order to adapt to multiple situations: The tuning reference, the number of harmonics and peaks to consider, the parameters used to create the *note dictionary* in NNLS chromas, the number of MFCCs to retain, etc. Recalculating the optimal parameters for different music styles, instrumentation and possible sonorities become impractical for many applications. With the arrival of deep learning (DL), researchers switched from manually designing these algorithms and tuning their parameters to letting neural networks (NN) learn them from examples. We call this set of chroma extraction algorithms *deep chromas*.

Most of the chromas extracted using these techniques were designed to improve the accuracy of ACE systems, mostly because a critical amount of data was available for western pop music, making it possible to train deep learning systems. In their work, Korzeniowski & Widmer [12] designed a system named the *deep chroma extractor* (DCE) that learns to extract chromas from the output of a filterbank. This is to the best of our knowledge the very first deep chroma, and its architecture is based one of the earliest kinds of NNs, the multilayer perceptron. Given a set of audio signals and corresponding chord labels, specific chroma targets are set up that contain activations of those pitch classes that correspond to the annotated chord at that time instance. For the estimation of a single chroma instance, the authors consider a context window of approximately 0.7 seconds around the time instance. This provides the neural network the opportunity to take preceding and subsequent content into account when estimating each chroma.

Another algorithm that extracts chroma features from audio for the purpose of ACE was proposed in [13]. Instead of using a fully connected layer and context frames, the authors employed convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The CNN part learns convolution kernels to convolve with the input spectrogram, looking for patterns related to pitch sensations. RNNs are a particular type of neural network that can consider information about previous and posterior audio frames. Here, the RNN part learns to ignore spurious changes in the spectrum through time, eliminating the necessity of using fixed-length context frames as in the *deep chroma extractor*.

2.3 Deep Chromas: Multi-pitch Labels

Since the deep chroma approaches of the previous section are trained on chord-based chroma targets, those systems will usually output a set of concurrent pitches. This might be acceptable or even desired for ACE applications, however, it makes those extractors less suitable for applications that rely on the analysis of more detailed pitch-class information. To obtain a more accurate representation, instead of using chord labels, Wu & Li [11] employ chroma targets based on note annotations, i.e. onset time, offset time and pitch. This enables the network to capture only the active pitch classes at each point in time. This network is based on a CNN. It uses the harmonic constant Q transform (HCQT) [14] as its input representation, which associates each CQT bin with its corresponding harmonics. This allows the network to see fundamental frequencies and harmonics simultaneously. Due to the lack of sufficient real-world data with corresponding note annotations, the authors revert to synthesized MIDI data. 6000 MIDI files were collected from the RWC Classical, Jazz and Genre dataset [15], and the Lakh MIDI dataset [16] ensuring a diverse range of musical styles. These files were synthesized using a sample-based SoundFont to create the audio input of the network. The problem of creating an audio dataset of real instruments with finer pitch-class information is that it would require note annotations including pitch, onset and offset times with sufficient precision. This laborious process does not scale easily to the amount of required data for training deep NNs.

Weiss et al. [17, 18] proposed to circumvent this problem by using real audio recordings of classical music together with non-aligned MIDI scores. To temporally match input and output data of the network, the authors employ the connectionist-temporal-classification (CTC) loss. The CTC was originally proposed for automatic speech recognition [19] where ground truth sentences require temporal alignment with speech utterances. This loss does not require a precise alignment between audio and score. Instead it only relies on the correct order of note events in both the MIDI score and the audio.

3. EXPERIMENTS

The chroma extractor families introduced in the previous section follow different design principles and partly serve

Polyphony	1	2	3	4	5	6
# Examples	1798	723	963	955	390	205

Table 1: Number of examples per polyphony level.

different purposes. We are interested in analyzing the behaviour of these systems for different types of audio input. More specifically, the accuracy of these systems was studied for audio inputs with varying levels of polyphony, as well as their ability to suppress non-tonal elements. For this purpose, we conducted a number of experiments in which we measured these properties quantitatively. A number of relevant use cases was defined and corresponding pairs of audio/chromagram were set up for each of those. To obtain audio/ground-truth chromagram pairs, we use a synthesized MIDI dataset with individual instrument stems (see Subsection 3.4). The audio of those pairs were then processed by a selection of existing chroma extraction algorithms. The output of each system was compared with the target chromas and metrics were computed to measure the similarity between actual and target chromas.

3.1 Varying Polyphony

In a first set of experiments, we evaluated the accuracy of the chroma algorithms for different levels of polyphony, i.e. different numbers of concurrent pitches. For that purpose, the dataset was divided into sections of 1, 2, 3, ... up to 6 concurrent pitches. We ensured that each fragment was at least 4 seconds long, resulting in several hundred examples for each polyphony level (see Table 1). We expect the chromas to contain high values at the present pitch classes and low values at all others. The accuracy of the systems is measured as the cosine similarity of each chromagram output $\mathbf{o}$ and the corresponding target chroma $\mathbf{t}$:

$$S(\mathbf{o}, \mathbf{t}) = \frac{\mathbf{o}^T \cdot \mathbf{t}}{||\mathbf{o}|| \cdot ||\mathbf{t}||} \tag{1}$$

These similarities cover a range from 0 to 1, with 0 indicating complete dissimilarity and 1 identical chromas. We call the cosine similarity between the algorithm's chroma and the ground truth *chroma accuracy*.

3.2 Suppression of Non-tonal Elements

A second set of experiments looked at the suppression of non-tonal components in the chroma output. As discussed in Section 1, chromas are expected to only capture the tonal content. Any non-tonal components should not contribute to the result. To measure the influence of percussive elements, we selected fragments from the datasets containing percussion with a minimum length of 10s. The individual stems of the dataset allowed us to store a version of each fragment without percussive elements alongside a version containing the full mix. The target chromas are the same for both cases since the percussive elements do not contribute to the targets. Again, we measure the cosine similarity between the actual and the target chromas. Percussive components are very common, particularly in popular music tracks, however, those elements usually have limited durations and hence only affect a part of the spectrogram. Stationary noise, on the other hand, poses a different challenge, covering a much wider time-frequency range and is also present in otherwise silent sections. Therefore we also evaluated the chroma extractors on the input sequences with added white noise instead of percussion. The ratio of tonal and non-tonal components depends on the mix and is most likely not fixed across music tracks. We hence evaluated percussion and noise at various levels of intensity. We use the signal-to-noise ratio (SNR) to characterize how present they are in relation to the tonal elements. Given a signal x with the tonal elements, and the signal y with only non-tonal elements, we define our SNR as:

$$SNR_{dB} = 10\, log_{10} \left(\frac{\sum_i x_i^2}{\sum_j y_j^2} \right) \tag{2}$$

In this case, we compare the algorithm's output for the signals x and y. Note that by varying the intensity of the non-tonal elements, we do not want to compare how much the algorithm resembles the ground truth, but how much of these elements' presence is able to achieve a significant change at the resulting chromagram. Instead, as we compare how similar is the resulting chromagram with and without the presence of non-tonal elements, we call in this case the result of the cosine distance *chroma similarity*.

Algorithm	Type	Name
Ellis & Poliner	Knowledge-based	Ellis
HPCP	Knowledge-based	Gomez
NNLS	Knowledge-based	Mauch
DCE	Deep: chords	Korzeniowski
McFee & Bello	Deep: chords	McFee
Wu & Li	Deep: multi-pitch	Wu
Weiss & Peeters	Deep: multi-pitch	Weiss

Table 2: The algorithms used in our experiments, along with the type of algorithms. The name column indicates the way we will refer to these algorithms in the results.

3.3 Tested Systems

We selected seven existing chroma extraction systems. An overview can be found in Table 2.

From the knowledge-based chromas (see Section 2.1) we employed Ellis & Poliner's system [6] as implemented in Librosa [20] [1] . For HPCP by Gómez we used the Essentia library [21]. In the case of NNLS we used the original vamp plugin [2] with a python wrapper; note that this algorithm returns a chroma for bass frequencies and another one for treble frequencies. We only used the treble chroma output of this algorithm.

For the family of chromas based on chord labels (see Section 2.2), we selected two algorithms. The DCE by Korzeniowski & Widmer as implemented in the Madmom library [22], which partially differs from the original model but according to the authors achieves similar results. For

[1] https://librosa.org/doc/main/generated/librosa.feature.chroma_stft.html

[2] http://www.isophonics.net/nnls-chroma

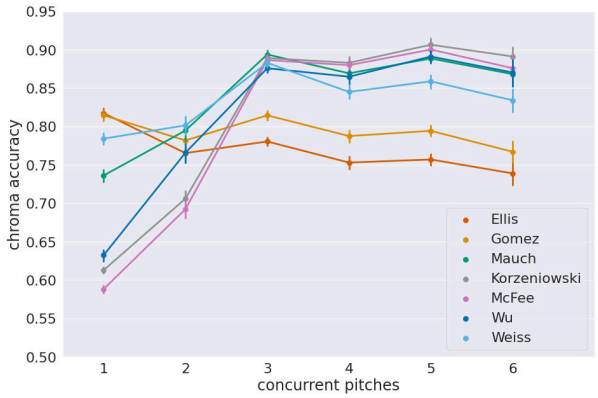


Figure 3: Mean chroma accuracy for each algorithm according to the number of pitches present. The algorithms that perform better for smaller levers of polyphony (Ellis and Gomez) perform worse at higher levels and viceversa.

the algorithm by McFee & Bello [13], we used the the public implementation provided by the authors [3].

For the family of deep chromas based on multi-pitch labels (see Section 2.3) we selected two algorithms: Wu & Li [11], and Weiss et al [18] with the public implementations given by the authors [4] [5]. Note that from the two works previously mentioned by Weiss, we selected the one using prealignment since it seems to provide slightly better results.

Each of the chromas based on deep learning operate at their own hop size, FFT size, and sample rate. For all the knowledge-based algorithms we used a hop size of 4410, a FFT size of 8096, and a sample rate of 44100.

3.4 Dataset

For all experiments, the Slakh dataset [23] was employed, which contains audio synthesized from 2100 files in the MIDI Lakh dataset (140 hours in total). There are 34 different instrument categories that cover a wide range of musical instrument timbres. The MIDI files were used to set up our target chroma representations for each track. The target chromas were sampled with the same hop size as the algorithms.

4. RESULTS

4.1 Varying Polyphony

The results for the polyphony experiments can be seen in Figure 3. This figure shows the mean chroma accuracy for the different levels of polyphony described in section 3.1.

We can observe two main tendencies: Chroma algorithms with less intricate pattern matching mechanisms such as Ellis or Gomez, perform better at lower levels of polyphony, but achieve lower chroma accuracies as the number of concurrent pitches grows. The rest of the algorithms on the other hand, perform worse for 1 or 2 concurrent pitches,

[3] https://github.com/bmcfee/crema

[4] https://github.com/Xiao-Ming/ChordRecognitionMIDITrainedExtractor

[5] https://github.com/christofw/pitchclass_mctc

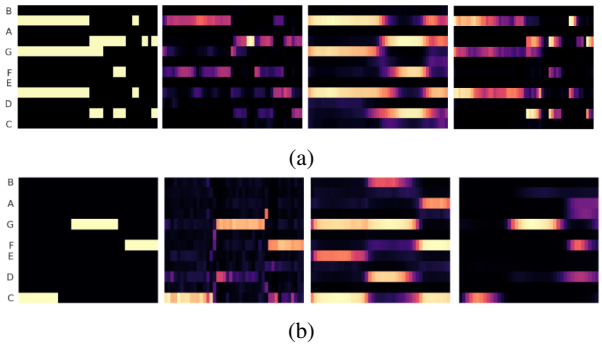


Figure 4: From left to right, the target chromas and chromagrams from Gomez, Korzeniowski, and Weiss.

but then reach a high chroma accuracies for 3 or more concurrent pitch-classes.

Deep chromas based on chords are at the lower part of this chart for 1 and 2 concurrent pitches. This is an expected behaviour since the models were trained on chord labels that contain three or more pitches, making the outputs of these algorithms usually chord-like chromas.

Deep chromas trained on multi-pitch labels follow the same trend as those trained on chord chromas: they exhibit better results for higher polyphony. But while the algorithm of Weiss is almost as good as Ellis or Gomez for 1 and 2 pitches, the chroma accuracy of Wu is significantly lower. We hypothesize that this could be attributed to the datasets used to train the models. Wu was trained with synthesised MIDI files from various datasets which consist mostly of pop music. This results in just a few passages where there are just one or two notes being played simultaneously. In contrast, Weiss was trained with classical music, which is more likely to have solo passages or sections where several instruments playing in unison.

To provide some intuition for these result, Figures 4a and 4b show chromagrams for two example sequences. The first is an excerpt of multiple concurrent pitches. The algorithm from Gomez gets some of the pitches right but struggles to clearly show all simultaneous notes; The DCE by Korzeniowski shows concurrent pitches more clearly, but at the same time smoothes the activations over time, resulting in 'chord-like' activations. The algorithm by Weiss generally shows less clear activations than the DCE but overall captures the finer details of the target chromagrams while at the same time recognising concurrent pitches. The second example in Fig. 4b shows a short melody with only a single active pitch at each time instance. Gomez' algorithm clearly highlights the correct pitch classes, but also contains spurious peaks, most likely caused by overtones that were not correctly assigned to their fundamentals. The DCE by Korzeniowski on the other hand contains activations for pitches that are actually not present in the audio at all. In order to produce chord-like chromas, the additional pitches form a major triad above the actual pitch. The extraction algorithm by Weiss only contains spurious activations for the last note.

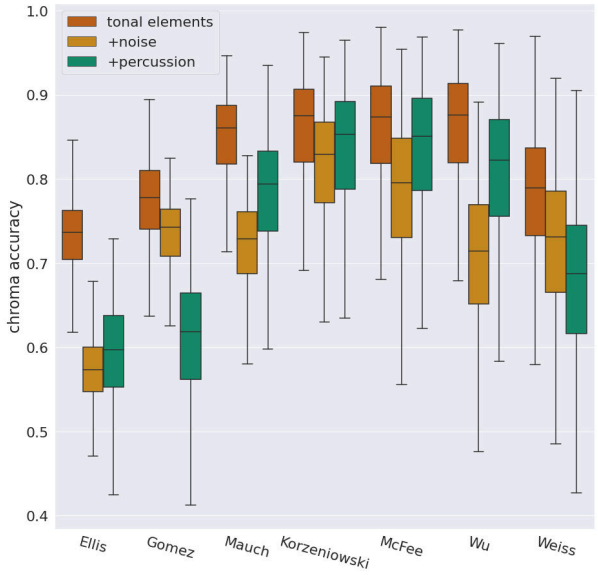


Figure 5: Chroma accuracy in 3 different scenarios: Just tonal information, added noise, and percussion. The algorithms performed best when only tonal elements were present in the signal. After adding noise or percussion ($SNR_{dB} = -15$) the output of the chromagrams resembled less to the targets.

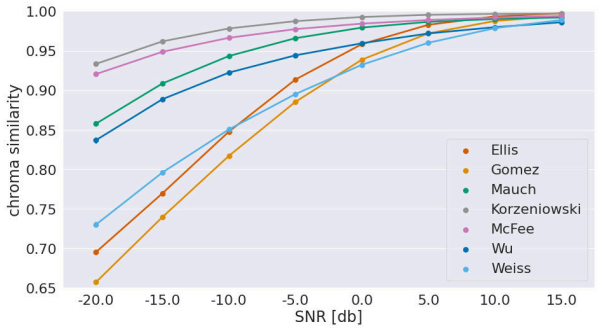


Figure 6: Chroma similarity as a function of percussion intensity. A smaller slope means that the algorithm is less affected by the presence of percussion.

4.2 Suppression of Non-tonal Elements

Figure 5 shows the results of the experiments with non-tonal components. It can clearly be seen that the addition of noise and percussion in all cases degrades the accuracy of the chromas. While the algorithms by Korzeniowski, McFee and Wu are least affected by the presence of non-tonal elements, all other algorithms show significant losses in accuracy. However, noise and percussion do not affect the algorithms in the same way. The peak selection from Gomez for example deals very well with noise, however, it struggles to suppress percussive elements in the spectrum. The opposite is the case for the algorithms by Mauch and Wu: noise seems to affect these algorithms more than percussion.

Figures 6 and 7 show how the intensity of the non-tonal elements (percussion and noise, respectively) affects the

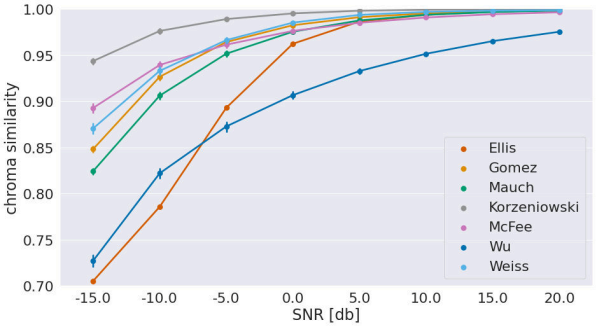


Figure 7: Chroma similarity as a function of noise intensity. A smaller slope means that the algorithm is less affected by the presence of noise.

algorithms. Some algorithms were able to deal with the non-tonal elements even with an $SNR_{dB} = -15$, and after $SNR_{dB} = 0$ most of the algorithms had a chroma similarity of 0.95 for both scenarios. Notice however that Wu is affected by noise even when this is quite small compared to the tonal signal.

5. CONCLUSIONS

In this work we investigated the performance of several existing pitch chroma extraction algorithms, analyzing their capability to deal with tonal content of increasing polyphony, as well as their robustness against non-tonal components. Given a dataset of music tracks with corresponding MIDI ground truth, pitch chromas were extracted by each algorithm for each scenario, and their accuracy w.r.t. a target chroma representation was measured. Results showed that chroma extraction algorithms based on deep learning produce a more accurate representation for higher polyphony levels and are generally more robust in suppressing non-tonal components. However, their reduced accuracy for lower polyphony levels hints at biases in the corresponding training sets.

Chromas are a well known and widespread feature in MIR. We argue that a generic chroma extraction algorithm should capture the tonal content as it is, thereby neither adding pitches that are not present in the audio nor performing additional processing steps such as temporal smoothing. While this might be obvious for applications such as audio-to-score alignment for which a temporal resolution at the note level is required [3], it will also be beneficial for ACE as it disentangles the detection of active pitch classes from the interpretation by a musical model.

Our results in Fig. 5 show that chroma extraction algorithms overall yield meaningful representations with decent accuracies. However, even for the case of music containing only tonal components, median accuracies do not exceed the 90% mark. With additional percussion or noise, these accuracies decrease. This shows that there is still room for improvement in the overall quality of the algorithms. The fact that the algorithm by Wu & Li achieves the highest results for content with only tonal components, encourages us to think that using multi-pitch content for the training of chroma algorithms is indeed worthwhile and

might pave the way to more accurate chroma representations.

A crucial factor for the training of better systems, however, is the choice of training data. Obviously, a sufficient amount of real-world audio with precise annotations is required, but also other qualitative data properties seem to be important: a diverse number of timbres and the presence of unpitched percussion and potentially non-musical sounds. Biases in the number of concurrent pitches should be addressed as well. This could be either by balancing the levels of polyphony in the training data or using loss functions that can diminish the number of false positives, such as the Weighted Binary Cross Entropy [24].

Acknowledgments

This research took place as part of a collaboration between the Huawei Munich Research center and the Pompeu Fabra University in Barcelona.

6. REFERENCES

[1] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Arxiv*, 2016.

[2] F. Yesiler, J. Serra, and E. Gomez, "Accurate and scalable version identification using musically-motivated embeddings," in *International Conference on Acoustics, Speech and Signal Processing*. IEEE, may 2020, pp. 21–25.

[3] C. Joder, S. Essid, and G. Richard, "A comparative study of tonal acoustic features for a symbolic level music-to-score alignment," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 409–412.

[4] G. Bernardes, M. Davies, and C. Guedes, "A hierarchical harmonic mixing method," in *Music Technology with Swing*. Springer International Publishing, 2018, pp. 151–170.

[5] T. Fujishima, "Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music," in *International Computer Music Conference Proceedings*, vol. 9, no. 6, 1999, pp. 464–467.

[6] D. P. Ellis and G. E. Poliner, "Identifying 'cover songs' with chroma features and dynamic programming beat tracking," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, 2007, pp. IV–1429–IV–1432.

[7] E. Gómez, "Tonal description of music audio signals," Ph.D. dissertation, University Pompeu Fabra, Barcelona, Spain, July 2006.

[8] M. Mauch and S. Dixon, "Approximate note transcription for the improved identification of difficult chords," in *International Society for Music Information Retrieval Conference*, 2010, pp. 135–140.

[9] M. Muller and S. Ewert, "Towards timbre-invariant audio features for harmony-based music," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 649–662, mar 2010.

[10] H. Terasawa, M. Slaney, and J. Berger, "The thirteen colors of timbre," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE, 2005, pp. 323–326.

[11] Y. Wu and W. Li, "Automatic audio chord recognition with MIDI-trained deep feature and BLSTM-CRF sequence decoding model," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 2, pp. 355–366, feb 2019.

[12] F. Korzeniowski and G. Widmer, "Feature learning for chord recognition: The deep chroma extractor," *International Society for Music Information Retrieval Conference*, pp. 37–43, 2016.

[13] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition." in *International Society for Music Information Retrieval*. Zenodo, 2017.

[14] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for f0 estimation in polyphonic music." in *International Society for Music Information Retrieval*. Zenodo, 2017.

[15] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "Rwc music database: Popular, classical, and jazz music databases," in *International Conference on Music Information Retrieval*, 10 2002, pp. 287–288.

[16] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, Columbia University, 2016.

[17] C. Weiss, J. Zeitler, T. Zunner, F. Schuberth, and M. Müller, "Learning Pitch-Class Representations from Score- Audio Pairs of Classical Music," in *International Society for Music Information Retrieval Conference*. Online: International Society for Music Information Retrieval, Nov. 2021, pp. 746–753.

[18] C. Weiss and G. Peeters, "Training Deep Pitch-Class Representations With a Multi-Label CTC Loss," in *International Society for Music Information Retrieval Conference*. Online: International Society for Music Information Retrieval, Nov. 2021, pp. 754–761.

[19] C. Wigington, B. Price, and S. Cohen, "Multi-label connectionist temporal classification," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, sep 2019.

[20] B. McFee, C. Raffel, D. Liang, D. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference*, vol. 8. SciPy, 2015.

[21] D. Bogdanov, X. Serra, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, and J. Zapata, "ESSENTIA," in *Proceedings of the 21st ACM international conference on Multimedia - MM '13*. ACM Press, 2013.

[22] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proceedings of the 24th ACM international conference on Multimedia*. Amsterdam, The Netherlands: ACM, oct 2016, pp. 1174–1178.

[23] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019.

[24] J. Zeitler, "Extracting tonal features for music analysis using deep learning," Master's thesis, ASC Major Research Project, Friedrich-Alexander-University of Erlangen-Nuremberg, Erlangen, 2020.

A Robotic Drummer with a Flexible Joint: the Effect of Passive Impedance on Drumming

Seyed Mojtaba Karbasi
RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion
Department of Informatics
University of Oslo
mojtabak@ifi.uio.no

Alexander Refsum Jensenius
RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion
Department of Musicology
University of Oslo
a.r.jensenius@imv.uio.no

Rolf Inge Godøy
RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion
Department of Musicology
University of Oslo
r.i.godoy@imv.uio.no

Jim Torresen
RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion
Department of Informatics
University of Oslo
jimtoer@ifi.uio.no

ABSTRACT

This paper introduces a drum robot with certain mechanical specifications and analyze its capabilities according to the drumming sound results when applied on a regular snare drum. The robot has two degrees of freedom, actuated by one quasi direct-drive servo motor. The gripper of the robot features a flexible joint with passive springs. We report on an experiment in which we have looked at the drum roll performance by the robot while changing a few control variables such as frequency and amplitude of the motion. Both single-stroke and double-stroke drum rolls can be performed by the robot by changing the control variables. The results of this experiment lay the groundwork for developing an intelligent algorithm for the robot to learn musical patterns by interacting with the drum.

1. INTRODUCTION

What can be expected from a drum robot? How can robots be used to create novel music? These questions have been pursued by researchers in the field of musical robotics in recent years, and we have seen robotic systems designed to perform music with different objectives achieving significant results [1, 2]. With the advancements in artificial intelligence and robotic systems, new capabilities have been explored in this field. One major aspect of musical robots that can lead to the emergence of creative results is the ability to learn skills autonomously. To make it feasible, it is important to make the robot utilize its potential and mechanical capabilities to play a musical instrument.

A main challenge when designing musical robots is to find the best sound-producing mechanism. Many musical robots have been inspired by the way humans play instruments. However, the mechanical specifications of robots are completely different to the physical properties of the human body. Thus, almost all of the robotic arms designed to play drums contain a fixed gripper and they usually have less than three degrees of freedom. In comparison, a human arm has a much more complex mechanism, especially the hand and the fingers which hold the drumstick. A key question is whether a robot needs the same mechanical complexity to perform equally well as a human drummer? One can argue that if we want the complete capabilities of a human arm, we should consider the same mechanical attributes. However, for a specific task, like drumming, all of the physical complexity of the human arm may not be necessary. In fact, in the best case, a robot should perform most efficiently according to its mechanical characteristics, rather than just mimicking human musicians. For this purpose, it is important to utilize the maximum capabilities of a robotic system. This can be achieved by an analysis-by-synthesis approach for studying robotic drumming.

Human drummers learn by practicing how to use their hands' physical capabilities to perform efficiently. The

Figure 1: The robot prototype. The robot has two joints, one actuated by a servo motor and the other one contains passive springs to make the gripper flexible.

Copyright: © 2022 Seyed Mojtaba Karbasi et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

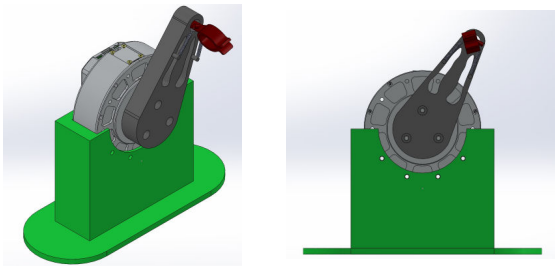
Figure 2: The drum robot playing a snare drum.

physical and mechanical constraints of the body shape the way a drummer learns the skills [3]. It is hard to build a robotic arm with the same mechanical properties as a human arm. A different approach, and the one taken in this paper, is to make a robot learn the necessary skills according to its physical constraints. In other words, when we have a robot with specific mechanical characteristics, it is expected that the robot should be able to learn the skills according to those constraints.

In this study, we introduce a drum robot which is designed and built using 3D-printing technology to play a snare drum (Figure 1). The robot has two degrees of freedom, with a passive flexible gripper and an actively controlled joint with a servo motor. In the paper, we present its design and construction and show how the flexible joint affects playing drum roll patterns. This can be seen as an analysis-by-synthesis approach to explore the drumming capabilities of the robot prototype.

2. RELATED WORKS

Several researchers have investigated the importance of mechanical actuation in drum robots. Hajian et al. developed a single-joint, variable stiffness robotic arm using pneumatic actuators in an agonist-antagonist arrangement [4]. Despite a successful demonstration of drum rolls, the authors acknowledge the slight possibility of their technology being implemented in practice due to servo delays in passive impedance control. Kim et al. use variable stiffness actuators (VSA) to reproduce single and double snare drum strokes, however, the device's utility is restricted by the VSAs' stiffness modification time [5]. Bretan et al. developed a robotic drumming prosthesis for an amputee drummer, controlled by electromyography (EMG) signals [6, 7]. Furthermore, Yang et al. used different actuation systems in order to achieve dynamic capabilities of the hand to perform multiple stroke drum rolls for the same robotic prosthesis [8]. They have also introduced a dynamical model that simulates the trajectory of the drumstick and estimates the control gains for a variety of natural bouncing patterns. A similar approach has been utilized in the Shimon robot, a robotic marimba player, to provide a wider dynamic range response compared to solenoid actuators [9].



(a) (b)

Figure 3: The 3D design of the drum robot.

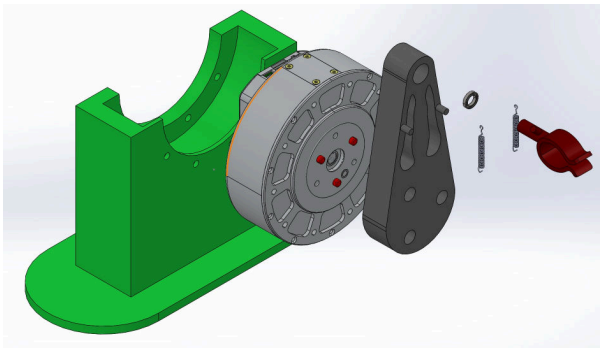


Figure 4: Each part of the robot used for prototyping the robot: base, servo motor, link, gripper, springs and ball bearing.

Other types of actuators have been used in percussion robots in recent years. For instance, Van Rooyen et al. used voice coil actuators for a percussion robot [10]. Another actuation system based on transducer, proposed by Brown and Topel, is used for achieving nonlinear acoustic synthesis effect for a snare drum [11]. Each of these actuation techniques leads to achieving certain targets in percussion performance.

The main characteristic of the robot introduced in this work is the two-degree of freedom mechanism with a flexible gripper. This provides for a more complex robot-drum interaction than what has been explored in previous studies. However, adding this complexity also makes it more difficult to develop control strategies for desired drumming tasks. We have previously proposed an interactive learning algorithm for adjusting the joint impedance of a drum robot to perform double stroke drum rolls [12]. That algorithm was based on simulations of a two-degree of freedom robotic arm, comparable to the robot introduced in this paper. Our previous results were based on simulation, while in this paper we present results from a physical robot.

3. THE DRUM ROBOT SYSTEM

The design of the robot, including the mechanical design of the body, control and actuation, is based on the natural specifications of the drumming of human drummers. The details of the designed system are described in this section.

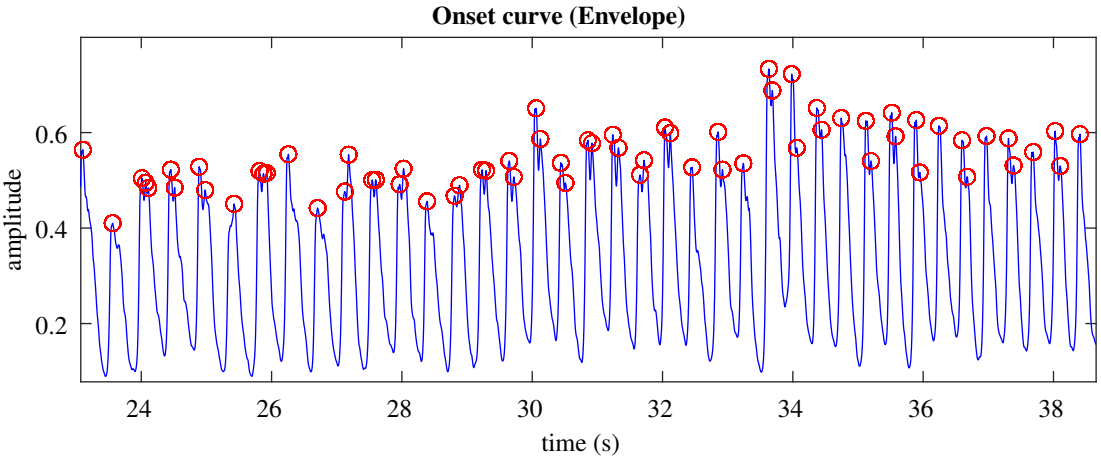


Figure 5: The amplitude envelope of a recorded drumming sound performed by the robot. The onset events are detected using the calculated envelope. The features used for analyzing the results are the relative amplitude of the detected events and time difference between consecutive beats.

3.1 Hardware

The robot hardware consists of 3D printed parts, passive springs and a quasi-direct drive servo motor (Figure 3). The design phase started by selecting a motor with the relevant characteristics for drumming: a Gyems RMD-X8. This motor is widely available in large quantities, relatively well documented and supported, and affordable at the time of writing of this paper. The RMD-X8 motor includes a gear ratio of 6:1 and CAN bus connectivity. It uses high resolution magnetic encoders and FOC drivers for working in different control modes such as position, velocity or torque control. This type of actuator is suitable for drumming tasks because of its ability to control torque as well as motion at high velocity (250rpm no-load speed). The torque control mode makes the robot capable of having impedance control of the arm. As we mentioned earlier, the mechanical impedance is an important property in drumming.

For building the arm, three main pieces were designed and 3D printed on a Fortus 250mc with ABSplus thermoplastic. The 3D model of the robot was designed using SolidWorks (Figure 4). We employed an iterative design strategy in which the model was continuously refined and tested until we had a prototype that was strong enough to handle the rebounding force during fast drumming while at the same time have a low enough weight to not limit the speed of the actuator.

The initial design of the gripper did not contain springs but a fixed gripper for the arm. As it turned out, the fixed gripper could not handle the rebounding forces in high-speed drumming. The result was numerous broken prototypes. This eventually led to adding flexibility to the gripper using a spring mechanism. The stiffness of the spring is a crucial parameter in the rebounding motion of the robot. With larger values of stiffness coefficient, the robot can play faster drum rolls. After some testing, we ended up with a spring with a stiffness coefficient of 25 N/m.

3.2 Software

The robot is controlled using an Arduino board, communicating with the motor via the CAN bus protocol. We developed a control library in C++ for performing drumming tasks using different control modes of the motor. In this study, a position-based control strategy is designed for the motor to track a sinusoidal trajectory with a desired frequency and amplitude to perform drum rolls with different frequencies and intensities.

4. EXPERIMENT DESIGN

For testing the physical capabilities of the robot, we designed an experiment to perform drum rolls. The purpose of the experiment was to find mappings between control variables of the robot motion and rhythmic features of the resulting drumming. This was based on extracting the *time difference* between consecutive beats played by the robot and their *relative amplitude*. These two features are essential in conveying the rhythm in drumming. The drum robot needs to play with precise timing and intensity in order to perform proper musical rhythms.

Control variable and parameters	Range
Amplitude of the motion	15-35 °
Frequency of the motion	1.5-5 *Hz*
Stiffness of the flexible joint	25 N/m
Controller frequency rate	300 *Hz*
Sound recording sampling rate	44100 *Hz*

Table 1: Value ranges of control variables and parameters used for the experiment.

We defined two variables in the control space: the frequency and amplitude of the robot's motion. We also defined two dimensions in the observation space: the time difference between consecutive attacks and their relative sound amplitude.

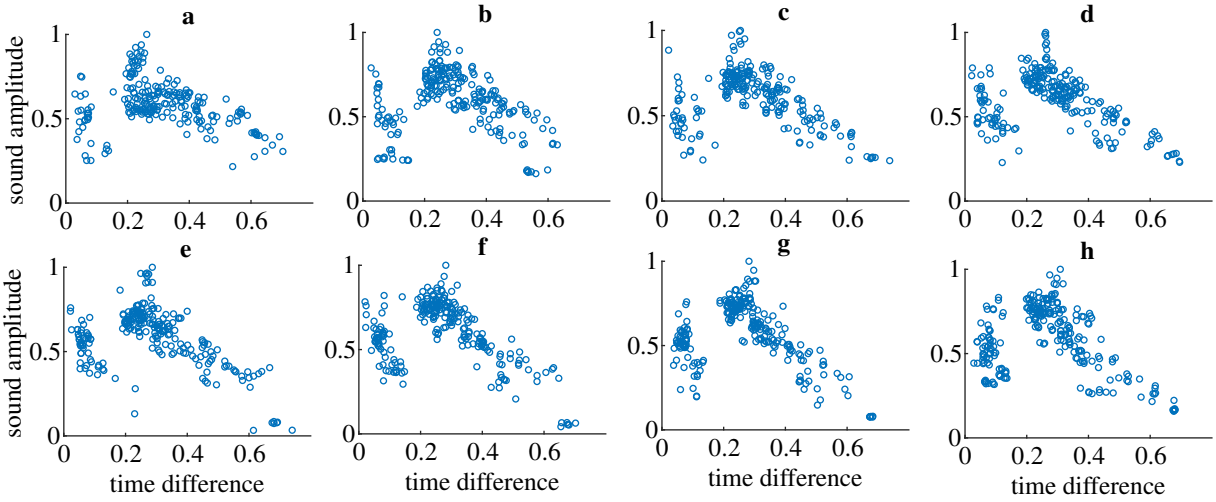


Figure 6: The extracted data points from the experiment in the observation space: time difference between consecutive notes (seconds) and relative sound amplitude. Each data set represents drum rolls with fixed amplitude of the motion and varying frequency between 1.5-5Hz and each data point represents a note played by the robot. Figure (a) has the smallest amplitude, and figure (h) has the largest amplitude.

The expimerntal settings are summarized in Table 1. For each experimental task, we used a fixed amplitude and increased the frequency of the motion within the range 1.5–5 Hz. We repeated the process for each amplitude value. The amplitude of the motion is defined by the angular range for the trajectory of the motor. The angular range of the amplitude of the motion varies between 15–35 °.

Sound features were extracted from the recorded sound files in Matlab using the MIR toolbox [13]. Onsets were detected using the amplitude envelope of the sound wave. A sample onset curve of the drumming sound is illustrated in Figure 5, showing the extracted events with red circles.

5. RESULTS AND DISCUSSION

As can be seen in the summary in Figure 6, similar patterns can be recognized in all of the recorded data sets. First, each data set consists of two distinguishable clusters. These clusters represent single stroke drum rolls (on the right side) and double stroke drum rolls (on the left side). This can more clearly be seen in Figure 7, which illustrates one data set (labeled (h) in Figure 6), in which single and double strokes are separated. Also in Figure 8, the impact of the changes in the frequency of the motion on the drumming motion can be observed.

Another general observation is the relationship between frequency and amplitude. By looking at the single strokes, we can see that when the frequency has been increased (smaller time differences), the sound amplitude also increases. It is natural to decrease the amplitude of the motion in faster drum rolls to maintain the same amplitude.

The data sets show how the observation space can be reached by changing the control variables. For instance, there are some “empty” areas in the observation space. This means that by adjusting the available control variables, performing a drum roll (with a specific frequency and amplitude) that resembles the empty areas is not feasible. In other words, the limitations and physical constraints of the robot determine the drumming result we can get by changing the control variables. This means that such an experimental approach can be used for the robot to explore feasible control possibilities to reach and recognize the reachable areas in the observation space. As mentioned before, the data can be used for learning algorithms that could find all the possible complex rhythmic patterns. This will allow the robot to utilize all its mechanical and physical potential.

6. CONCLUSION

In this paper, we presented a drum robot prototype and explored its ability to perform drum rolls. The main goal of the experiment was to investigate how drum robot performance is limited to its physical constraints. Furthermore, we are interested in exploring how a robot can find its optimum control parameters by performing at specific frequencies and amplitudes. The results are relevant for further iterations of the robot design and to develop algorithms for the robot to learn drumming tasks. There are some key points that can be discussed regarding the results:

- The passive spring used in the flexible joint limits the frequency of the double strokes. This means that for performing double strokes, the timing of the strokes is fixed regardless of the values of the frequency and amplitude. This issue can be solved by changing the trajectory offset or the point that the drumstick collides with the drum membrane. In this way, by adding an additional control variable, the feasible areas expand in the observation space for the robot, and the robot can perform with more variability.
- Having two degrees of freedom and being underactuated, the control of the robot becomes more challenging. The nonlinear behavior of the arm can not

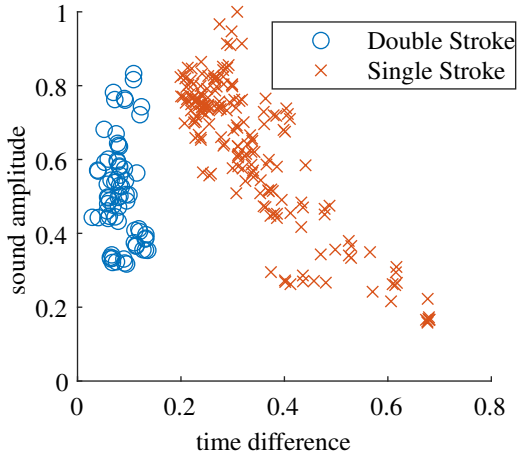


Figure 7: Single strokes and double strokes separated. The frequency of the double strokes is limited due to the presence of the passive spring in the gripper joint.

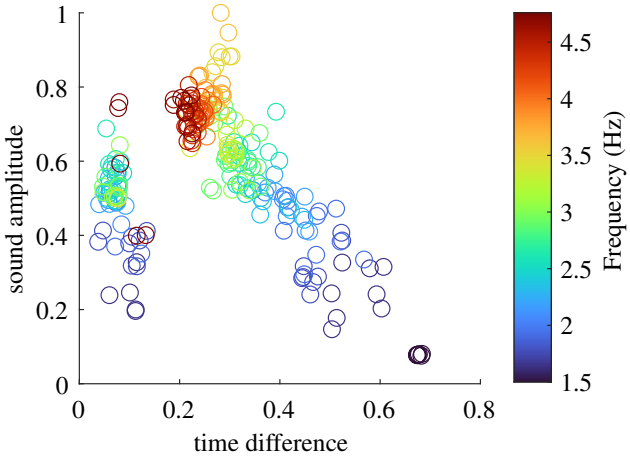


Figure 8: There is a direct relationship between the frequency of the motion and drumming features. The color of each point indicates the frequency of motion in the experiment.

be easily predicted. One way to develop satisfactory control parameters is by finding patterns in the drumming output. In addition to drum rolls, other complex rhythmic tasks can be explored by the robot without analyzing the dynamics of the robot and the drum. This model-free approach can be tried by using interactive intelligent algorithms like reinforcement learning to improve the creativity and efficiency of the drum robot. Some solutions might be completely different from how human drummers perform since the mechanics are different. Thus, it would be interesting to see the robot learn interactively.

In the future we will use machine learning methods to solve the nonlinear relationships between motion parameters of the robot and its drumming results. In particular, reinforcement learning has a big potential in training the control strategies of the robot. One important question when using reinforcement learning is how the objective for the robot can be defined, and how creative the behaviour can be. We will address both these questions in future studies.

Acknowledgments

This work is supported by the Research Council of Norway through its Centres of Excellence scheme, project number: 262762.

7. REFERENCES

[1] G. Weinberg and S. Driscoll, "Toward robotic musicianship," *Computer Music Journal*, pp. 28–45, 2006.

[2] M. Bretan and G. Weinberg, "A survey of robotic musicianship," *Communications of the ACM*, vol. 59, no. 5, pp. 100–109, 2016.

[3] R. I. Godøy, "Motor constraints shaping musical experience," *Music Theory Online*, vol. 24, no. 3, 2018.

[4] A. Z. Hajian, D. S. Sanchez, and R. D. Howe, "Drum roll: Increasing bandwidth through passive impedance modulation," in *Proceedings of International Conference on Robotics and Automation*, vol. 3. IEEE, 1997, pp. 2294–2299.

[5] Y. G. Kim, M. Garabini, J. Park, and A. Bicchi, "Drum stroke variation using variable stiffness actuators," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3892–3897.

[6] M. Bretan, D. Gopinath, P. Mullins, and G. Weinberg, "A robotic prosthesis for an amputee drummer," *arXiv preprint arXiv:1612.04391*, 2016.

[7] D. Gopinath and G. Weinberg, "A generative physical model approach for enhancing the stroke palette for robotic drummers," *Robotics and autonomous systems*, vol. 86, pp. 207–215, 2016.

[8] N. Yang, R. Sha, R. Sankaranarayanan, Q. Sun, and G. Weinberg, "Drumming arm: an upper-limb prosthetic system to restore grip control for a transradial amputee drummer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 317–10 323.

[9] N. Yang, R. Savery, R. Sankaranarayanan, L. Zahray, and G. Weinberg, "Mechatronics-driven musical expressivity for robotic percussionists," *arXiv preprint arXiv:2007.14850*, 2020.

[10] R. Van Rooyen, A. Schloss, and G. Tzanetakis, "Voice coil actuators for percussion robotics." in *NIME*, 2017, pp. 1–6.

[11] H. M. Brown and S. Topel, "Drmmr: An augmented percussion implement." in *NIME*, 2019, pp. 116–121.

[12] S. M. Karbasi, R. I. Godøy, A. R. Jensenius, and J. Tørresen, "A learning method for stiffness control

of a drum robot for rebounding double strokes," in *2021 7th International Conference on Mechatronics and Robotics Engineering (ICMRE)*. IEEE, 2021, pp. 54–58.

[13] O. Lartillot and P. Toiviainen, "A matlab toolbox for musical feature extraction from audio," in *International conference on digital audio effects*, vol. 237. Bordeaux, 2007, p. 244.

Constructive Accumulation: A Look into Self-Organizing Strategies for Aggregating Temporal Events along the Rhythm-Timbre Continuum

Nolan Lem
Center for Computer Research in Music and Acoustics (CCRMA) Stanford University
nlem@ccrma.stanford.edu

ABSTRACT

This paper summarizes several algorithmic techniques taken from dynamical system modeling that can repurposed to produce 'constructive accumulation', a paradigm for describing the process by which concurrent, rhythmic events or sound sources gradually converge over time. Using different behaviors associated with self-organization, these strategies employ networks of locally and globally coupled oscillators that can generate a range of swarm-like acoustic textures, each reflecting a range of synchronous states along a rhythm-timbre spectrum. More specifically, I review the behavioral dynamics and clustering kinetics of 'scrambler' oscillators, pulse-coupled oscillators, and a group of Kuramoto oscillators with feedback. I also provide a basis for sonifying such models, briefly touching on concepts in psychoacoustics that are relevant to the design of such sonic systems. Lastly, I demonstrate the way in which they're implemented, controlled, and composed for as generative models in my recent artwork, 'In Praise of Idleness'.

1. INTRODUCTION

While synchronization in music performance is evidence of our own dynamic ability to synchronize to beat in musical settings, several groups of animal and insect populations have been shown to resort to similar modes of mutual entrainment in the form of chorusing, coordinated signaling, and stridulation [1] [2]. While not an explicitly auditory phenomenon, the prominent paradigm of firefly signaling is a classic example that illustrates the extent to which locally interactive behavior can result in emergent, global patterns [3]. In urban and metropolitan life, the sonic world is awash with noisy, periodic sounds such as public transportation, the harsh, repetitive sounds of a construction site, or the nervous energy of restless legs under a desk. Navigating, interacting with, and understanding our physical environment oftentimes entails using auditory cues in order to guide our behavior [4] [5]. As in music, we take note of the temporal relationships between regularly occurring events in order to infer information about its source and make predictions about the future [6] [7]. However what qualities of the sounding environment allow us to infer that they are indeed connected?

Research has shown that when sonic events occur with enough regularity and are spaced close enough in time, we tend to perceive them as coming from a similar temporal structuring process [8] [9]. However, this becomes more complicated when we are tasked with unpacking these dynamics in the context of simultaneous sound sources that may be defined by global synchrony, a term used to describe a continuum of dynamic states that reflect how coordinated, aligned, or synchronized individuals are in a group or system. Here I define a process called *constructive accumulation* that aims to describe any time-based, sonic environment or process that sees the gradual aggregation, or accumulation, of seemingly disparate sound events converging over time into a temporal order or pattern. This broad description is applicable to stridulating cricket population, groups of mechanically coupled metronomes, members of an orchestra, and the applause that follows a performance of the latter, all examples that have been modelled using synchronization strategies discussed in this paper [10] [11].

It's worthwhile to take a moment and try to unpack what is meant by 'order' or 'pattern' as it applies to rhythmic, auditory signals, a line of inquiry that has been examined in analyzing interaction paradigms in biomusicological investigations [12]. Here of course we are dealing with human perception and consequently the evaluation of subjective, qualitative criteria of phenomena that self-organizes in some way over time. With this in mind, I examine three models of collective behavior often analyzed in dynamical systems modeling in order to characterize such constructively accumulated processes as potential sources for sonic generation and analysis. Next, I provide a brief look at some of the relevant psychoacoustics research that may govern our perception of such auditory phenomena. Lastly, I present how these specific self-organizing models influenced the design and composition of one of my recent kinetic-sound installations.

2. MATHEMATICAL DESCRIPTION

2.1 Pulse-Coupled Oscillators

Pulse-coupled oscillators have been used to describe a number of mutually synchronous systems in the biological world (pacemaker cells, cricket chirping, insulin secretion in pancreas, menstrual cycles) [13] [14]. Pulse-coupling often relies on a coupling function that oversees

Copyright: © 2022 Nolan Lem et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

how coupling is applied when an oscillator 'fires' or 'triggers'. Phase response curves (PRC) govern this coupling relationship, parameterized by a coupling coefficient, as different PRCs shapes result in different dynamics. Here I define one of the simplest types of pulse-coupling among 'integrate-and-fire' oscillators that were initially inspired by the self aligning nature in pacemaker cells [15].

Using the model as defined Mirollo and Strogatz, a system of N pulse-coupled, integrate-and-fire (IF) oscillators, individual oscillators dynamics is defined in Equation (1), (1.1), and (1.2) [16].

$$\dot{x_i}(t) = F(x_i(t)) \tag{1}$$

$$\dot{x_i}(t^+) = 0, \text{ if } x_i(t) = 1 \tag{1.1}$$

$$x_i(t) = min(1, x_i(t^-) + m\epsilon) \tag{1.2}$$

The set of $(x_i)^N$ are the state variables in a system of N integrate-and-fire (IF) oscillators where the $\dot{x}$ notation implies the time derivative. Equation (1.1) shows how each oscillator's state variable is reset to 0 upon reaching the threshold of 1 (upon the next time interval, t^+). Also upon reaching this threshold, it is said to "fire" which describes how it modifies the state variable of the other oscillators in the group as shown in (1.2). The strength of this is dependent on the coupling coefficient, ϵ, which is distributed to other oscillators upon firing where t^- implies the state of the variable just before the oscillator has reached threshold. Assuming the i^{th} oscillator has not yet crossed the threshold, m is a function of the other oscillators that have also fired within the time interval and therefore the strength of each oscillator's state variable advancement is dependent on the number of oscillators firing within the same time interval.

Equation (2) shows how the state variable, x, is related to a phase variable, θ, which simplifies the analysis such that we can examine the system in terms of the phase variable and a function that relates θ to x.

$$x_i = f(\theta_i), f : [0, 1] -> [0, 1] \tag{2}$$

Equation (3) and (4) illustrate the function, $F(x)$ and $f(\theta)$, in terms of the state variable, x, and the phase state variable θ.

$$F(x) = \frac{e^b - 1}{b} e^{-bx}, b > 0 \tag{3}$$

$$f(\theta) = \frac{1}{b} ln(1 + (e^b - 1)\theta) \, , b > 0 \tag{4}$$

Now, we can a define a new function, f, that maps the phase state variable, θ, and following the convention of Mirollo and Strogatz is defined to be monotonically increasing and concave down. Equation (1.1) shows how the oscillator is coupled to the others from the coefficient, ϵ. b is a parameter that defines the relative concavity (down) of the function, f.

More simply, each oscillator rises toward a threshold value of 1, upon which it fires and is reset to 0. Upon this firing, other oscillators' phase states are either advanced by an amount, depending on the strength of ϵ, and/or pulled into the threshold for firing, whichever is less due to the min function in (1.2). The oscillators are globally coupled (fully connected) and maintain identical dynamics insofar as their phase state variable is governed by a function that controls its time-course, increasing monotonically (concave down) toward the threshold. The function constrains the phase state: it increases the phase state to 1 whereby it is then reset to 0.

2.1.1 Dynamics

IF pulse-coupled oscillators has been shown to eventually lead to full synchrony regardless of N and for all initial conditions and natural period if coupling is applied. There is evidence to suggest that they synchronize at times proportional to the log of the number of oscillators in the network. These outcomes were a result of global, all-to-all coupling; if we allow for coupling to only occur 'locally', for example in a chain, the time needed for synchronization decreases considerably, becoming a nearly linear function of the natural period and the coupling strength [17]. Despite the longer time scale when the natural period is small and when N is large, synchrony is still inevitable and the unusual synchronizing dynamics during this build-up period are still a subject of active research.

If we allow the natural frequencies of the oscillators to take on different values, synchrony is still achievable if they are kept within bounds. A number of papers have demonstrated how frequency mismatch and delays can lead to unusual dynamical regimes, some of which contain resonances in polyrhythmic relationships to its intrinsic frequency distributions [18].As will be shown in a later section, these types of oscillators settle into synchrony at a much more progressive, gradual rate than the spontaneous synchronization associated with continuously coupled systems. This has a number of desirable characteristics in terms of potential for sonic generation.

2.1.2 "Scrambler" Oscillators

Networks of scrambler oscillators probably are the most basic form of a loosely coupled oscillator system from which synchrony emerges through mean-field approximation [19]. A group of scrambler oscillators of size N are comprised of pulse-coupled oscillators each containing a phasor state variable, ϕ, that increases linearly and identically from a baseline of 0 to a threshold value of 1. Upon reaching the threshold value, the oscillator fires and performs three actions: 1) "scrambles", all other oscillators (and other extant 'clustered' oscillators) to a new random state. Oscillators already contained within a 'cluster' get reset to the same state. 2) oscillators with states within the threshold and 1/N of the threshold are subsumed, or absorbed, into the firing oscillators cluster. 3) the oscillator that fired (and any oscillator that just clustered along with it) is reset to the baseline state of 0. Lastly, if a cluster of oscillators reaches the threshold, then the condition for absorbing other oscillators becomes (threshold – j/N) where j is the number of clusters in that group. This has the effect

```
for i in iterations do
    for n in N do
        osc[n].state = osc[n].state + T
        if osc[n].state ≥ 1 then
            for j in N do
                if osc[j].state ≥ 1–(osc[n].cluster.size/N) then
                    osc[j].cluster = osc[n].cluster
                else
                    if osc[j].state ≥ (1 − (1/N)) then
                        osc[j].cluster = increment cluster counter
                osc[n].state = 0
            else
                osc[n].state = random(0,1-(1/N))
```

Table 1. Pseudocode for scrambler oscillation. NB: osc[n].cluster.size refers to the number of clusters that are tagged with the same counter number, not the size of the array itself.

of allowing more oscillators to become absorbed a single cluster grows in size.

Using the description initially defined by O'Keefe et al. (2015), I provide a brief pseudocode (see Table 1) rather than a formal analytical description since this type of system is better described as an iterative process. All of the N oscillators are initialized with a different integer representing their cluster state (to group oscillators into cluster groups once they've been absorbed into a cluster) and a different randomly initialized phase state. T is the increment that sets the natural period of oscillation.

2.1.3 Dynamics

O'Keefe et al. define a natural disorder parameter that provides a measure of system's 'fragmentation' based on how many clusters are present at any given time [19]. The system starts out maximally fragmented because each oscillator is only assigned to its own cluster group. The authors derive an expression for the rate equation that illustrates how the fragmentation decreases exponentially over time (with the natural disorder parameter reaching 1/N) as more oscillators are absorbed to synchrony. In general, scrambler oscillators achieve synchrony in a much less 'productive' way as compared to simple IF coupled oscillators because oscillators outside of threshold are simply reset to random values rather than being pulled up or down according to a PRC. Oscillators are pulled into synchrony without maintaining ordering since they are reset to a random value, unlike in pulse-coupling.

2.2 Summary Statistics: Phase Coherence

For any collection of phase states, we can also look at other summary statistics that provide a useful indication of the group's synchrony. These values are the phase coherence, R, also known as a circular mean vector and the average angle of the phase coherence, ψ. Mapping each phase state (0-1) of the oscillators above onto a circle (0-2π), we can derive an expression that relates the relative 'spread' or dispersion of the swarm of phases of each oscillator to a number between 0 and 1 with larger numbers associated with more synchrony. We define the phase coherence in Equation (5) where j is a complex number.

$$Re^{j\psi} = \frac{1}{N}\sum_{i=1}^{N} e^{j\phi_i} \tag{5}$$

Figure 1 shows plots from a numerical simulation of a group of 100 scrambler oscillators over a duration of around twelve seconds.

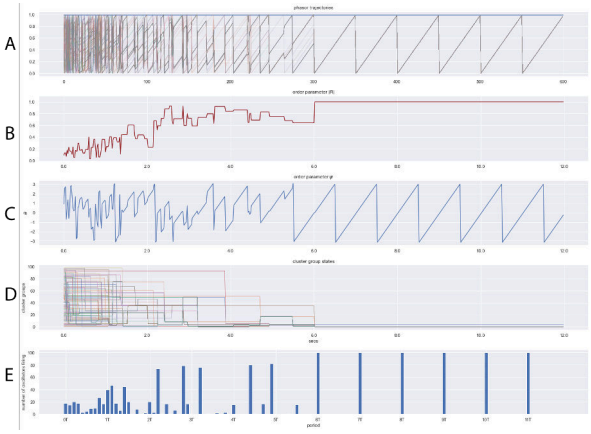


Figure 1. 100 Scrambler Oscillator Numerical Simulation. (Top to bottom) Individual oscillator phasor trajectories, Phase Coherence ($R(t)$), Average Angle ($\psi(t)$), Cluster Groupings, and Number of oscillators firing per natural period.

These plots illustrate the emergence of self-organization as the oscillators phases begin to align over a period of around six seconds. Plot A) shows the phasor trajectory of each of oscillator, Plot B) the phase coherence, $R(t)$, Plot C) the average angle, $\psi(t)$, and Plot D) the state variable of cluster groups for each oscillator over time. As more oscillators are recruited into synchrony, the number of distinct cluster groups decreases until every oscillators is in the same cluster. Lastly, Plot E) shows the number oscillators firing plotted in multiples of the natural period of the oscillators (as binned into one of ten temporal regions per natural period). Since each oscillator is identical, containing the same natural frequency, the entraining tempo that the oscillators will give rise to sync is predetermined by this initialization. Each plot provides useful visual information that allows us to gain insight into how the system self-organizes over time.

2.3 Continuous Coupling: Kuramoto Oscillators

My previous work has looked at Kuramoto oscillators from a number of different perspectives relevant for sound synthesis and music generation (for more information, please see [20] [21].Kuramoto oscillators are a type of limit-cycle oscillators with natural frequencies, ω_i, and a coupling term that continually adjusts their phases according to a phase response curve [22]. This last point is what primarily differentiates them from the previous two coupling algorithms that relied on "pulse coupling". The natural frequencies are typically drawn from different statistical distributions and since coupling is applied at all times, synchrony can result if coupling surpasses a critical coupling

value. The governing equation for a group of N Kuramoto oscillators is shown in Equation (6).

$$\dot{\phi}_{\mathrm{i}} = \omega_i + \frac{K_i}{N}\sum_{j!=i}^{N} sin(\phi_j - \phi_i) \tag{6}$$

Using the phase coherence from Equation (5), we can rewrite Equation (6) in terms of the complex order parameters, R and psi as shown in Equation (7) where an extra component, $\Lambda_e(\phi_i)$, is also added to allow for external forcing.

$$\dot{\phi}_{\mathrm{i}} = \omega_i + \Lambda_e(\phi_i) + \frac{K_i}{N} R \sum_{j=1}^{N} sin(\psi - \phi_i) \tag{7}$$

2.3.1 Dynamics

While explicit coupling is a function of the phase coherence, the external forcing function has been shown to modulate the time-varying mean field and thereby modulate the phase coherence over time which allows us to effectively 'tune' the synchrony of the system [23]. In this orientation, each oscillator interacts only with the mean-field approximated complex order parameter values, R and ψ. It's useful to imagine R and ψ in polar form where it's visually represented as a time-varying phasor, R(t)$\psi(t)°$, moving about a circle. $\psi(t)$ traces out the center of mass of the swarm of points, each point representing an oscillator, and $R(t)$ increases in length when the oscillators' phases are more aligned. The external forcing function, $\Lambda_e(\phi_i)$ modifies the the natural frequency distribution and allows for different modes of both forced and mutual entrainment [24].

If we let $g(\omega)$ be a normal distribution from which the oscillators' natural frequencies are drawn and allow to N go to ∞, the critical coupling–the value, K_c, for which the synchrony emerges–has been shown to be a function of the mean natural frequency, ω_c, found in the distribution ($K_c = 2/(\pi g(\omega_c))$. At the critical coupling, synchrony emerges very rapidly and spontaneously, often referred to as a first-order phase transition in which we see an abrupt jump from a low value of R to a large one.

Allowing the parameters, N, ω_i, K_i, and $\Lambda_e(\phi_i)$, to take on time varying values allows for a range of unusual semi-synchronous states to emerge including partial states of synchronization, frequency locking, chimeric states, bellepherone states (multiple coherent clusters), and resonant states [25] [26].

3. RHYTHMIC GENERATION SCHEMES

3.1 Oscillator Phase Mapping to Control Signals

The collective, self-organizing behaviors exhibited by these systems can be meaningfully applied to sound in a number of ways and similar topics have been explored in computer music research [27] [28]. Through relatively simple parameterization, we can exploit the system dynamics in such a way to produce a range of sonic outcomes that characterize constructive accumulation. This involves treating the dynamical systems as generative models that map the system output states to control signals that can be applied to sound production.

Through numerical simulation, we can plan, control, and compose for different outcomes depending on our musical intentions. Constructive accumulation deals specifically with the temporal organization of sound onsets; that is we are concerned with the building up of sonic mass using small, discrete sonic objects, a process that shares similarities in many approaches to computer music such as concatenative and granular synthesis, microsound, and compositional techniques involved in micropolyphony [29], [30], [31].

Circle maps are widely used in mathematics and physics to depict iterative phase states and have been employed in the service of sound synthesis in several ways [32]. Figure 2 illustrates such a circle map with several oscillators represented as black points, each moving about the circle with a frequency, $\omega_i = \dot{\phi}_i$. These control signals can subsequently be mapped to any number of musical parameters; the most simple form of this is to apply thresholds along the phase space and trigger a sound event once per cycle, when $\phi_i = 0$. However, extending this notion we can trigger multiple sound onsets during the course of a single limit-cycle to create a number of different rhythmic patterns. More formally stated: if we let, θ_m, be a set of M points along the unit circle, we create triggers when $\phi_i == \theta_m$.

3.1.1 Using mean field parameters

Similarly, we can use the complex order parameters, the phase coherence, $R(t)$, and the average angle, $\psi(t)$ as another control signal from which to assign to other control triggers. In this case, we can use the magnitude of the phase coherence and the angle of $\psi(t)$ trigger events based on a similar collection of m thresholded values defined as r_{tm} and ψ_{tm}. In this way, other control signals are only generated during specific periods of group synchrony or other dynamic regime of the system. The Figure shows two phase coherence magnitude trigger lengths ($r_{t1,t2}$), and two phase coherence angle trigger angles ($\psi_{t1,t2}$). Similarly, we allow the set of trigger points to take on different values over time in order to introduce more rhythmic variation into the system.

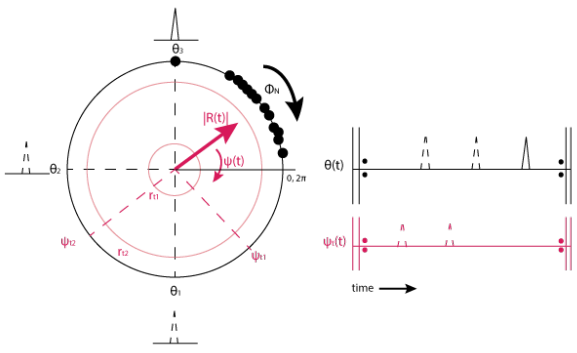


Figure 2. Circle map illustration of oscillator phase and complex order mapping to control signal triggers.

4. PSYCHOACOUSTIC CONSIDERATIONS: TIMBRE VS. RHYTHM FORMATION

Our ability to discern, process, and entrain to multiple, simultaneous sounds has been looked at from a number of perspectives in music cognition, psychoacoustics, and auditory scene analysis. Research has shown that auditory stream formation depends largely on the temporal coherence inherent in a sound signal [33]. While much of this line of research focuses on the separation and identification of simultaneous sound sources ('the cocktail party' problem'), the dynamical systems and generation scheme thus described are concerned with concurrent, quasi-periodic rhythmic sources, an auditory scene that has been less studied in sensorimotor synchronization studies [9]. More specifically, how does the simultaneous presentation of multiple, yet identical, periodic sound sources differ from attending to a normal acoustic environment that contains an abundance of non-coupled sounds?

Studies in sound texture perception provide one clue into how the brain encodes such auditory scenes comprised of similar acoustic events. McDermott and Simoncelli were able to realistically approximate soundscapes of relative temporal homogeneity (e.g. insect swarms, running water, and applause) by recreating the time-averaged statistics of an audio signal's frequency decomposition [34]. Since attending to every singular sources in such a texture would be cognitively demanding, this study supports the idea that when confronted with a large number of collective sound sources, we adapt to a temporal averaging strategy whereby timbre is identified through statistical inference. However, applying coupling to these systems, the resultant signals are no longer statistically 'stationary' and the emergence of cognitive processes implicated in beat extraction may take precedence. For example, one study looked at how 'embedded', repeated temporal-spectral acoustic features in mixtures have been shown to facilitate discernment and source identification [35].

Similarly, research in auditory scene analysis provides other clues. For instance, gestalt grouping and stream convergence based on temporal proximity has been shown to be a factor in how we perceptually merge different sounds with differing rhythms [36]. If enough onsets are temporally proximate, a signal's temporal envelope also provides auditory cues that allow for the emergent rhythm to be detected if the constituent elements are of similar, homogenous textures [37]. In developing his 'sound mass' pieces in the mid 20th century, the composer György Ligeti remarked how sounds produced around 18 times/second create the perception of a continuous texture, an observation also noted in Iannis Xenakis' writings [31]. In his music, global rhythms emerge from the local interactions of independent instruments, a compositional technique defined as a sort of micro-polyphony.

One unusual auditory environment that produces a similar spectrum of textures is the phenomena of synchronized clapping that occurs during audience applause. Several studies have examined the crowd dynamics using systems of coupled oscillators where they proposed that an audience will instinctually adapt their natural clapping frequency to a lower frequency which has the effect of reducing the frequency dispersion of intrinsic frequencies within the group [10].

My previous experimental research has looked at how we entrain to networks of coupled oscillators by examining how beat extraction is carried out when confronted with sounds generated in a similar schema [38]. In one study, participants were asked to tap to the 'beat' to auditory stimuli where coupling strengths were modified from 'weak' to 'strong'. Participants displayed a wide range of tap strategies that we classified as isochronous, patterned (non-isochronous but rhythmic), or dense (rapid tapping) with the latter two strategies associated more with weaker coupled sounds. This supports the notion that such auditory scenes encapsulate different understandings of sound through listening and interaction, perhaps through a form of active sensing.

These research questions motivate the need to carry out alternative approaches for reproducing these dynamics in artistic and aesthetically mediated environments. With this in mind, I present one such instantiation of these ideas in my installation, 'In Praise of Idleness'.

5. IN PRAISE OF IDLENESS

5.1 About the Piece

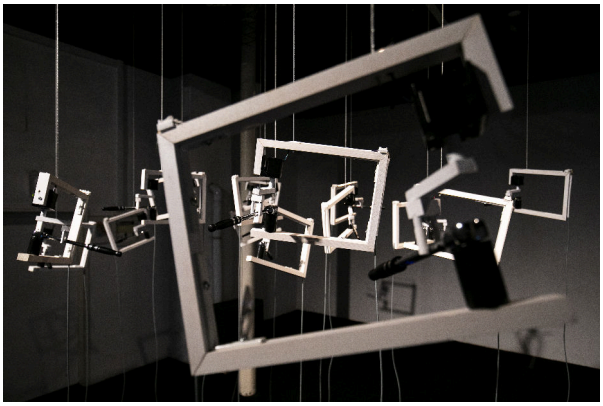


Figure 3. View of the Installation

In this site-specific installation [1] that premiered at Galerija Alkatraz in Ljubjlana, several large socket wrenches ("ratchets") are driven by an assembly of motors that kinetically rotate them in a number of coordinated ways. Mechanically, ratchets have an internal clicking mechanism that produces sound when it is rotated. As an audio-visual symbol linking mechanical labour with the passage of time, the socket wrenches evoke the ambient landscapes of swarms of stridulating insects, metronomes, or ticking clocks. In order to control their behavior, several different algorithmic techniques are used to choreograph the socket wrenches' collective dynamics, the result being a cooperative and sometimes crude synchronisation of movement and sound that is realized through their kinetic design and

[1] Please see: https://www.youtube.com/watch?v=68sq2jm_WCw&t=0s

through the "sound(s) of their own making", a reference to Robert Morris's piece of a similar title. Dialoguing with Jacques Attali's notion of noise as constituting "a political economy of music" this piece points to the ways in which simple repetition can be exploited to build up complex auditory textures through constructive accumulation.

5.2 Technical Requirements

The installation uses the rotational motion of 19 socket ratchets as a way to induce small mechanical clicks that serve as sound events using the circle map scheme shown in Figure 2. Each ratchet is housed in a wooden, c-shaped form with is hung from the ceiling. A small lever arm, attached to the shaft of a NEMA 17 stepper motor (0.59 N-m, 1.7A, 1.8°), rotates the ratchet when a pulsed voltage is applied to the stepper coils. Each stepper motor is controlled with a dedicated stepper motor driver (TBB6600) and powered with an individual power supply (12 V, 2 A). The required torque to drive the ratchet lever was calculated to be around 0.24 N-m.

An Arduino mega was used to control the rotation of the stepper motors using the Stepper Library and all the algorithms were written using the Arduino language. Numerical integration for the IF pulse-coupled, and Kuramoto oscillators using a forward Euler integration scheme which has been shown to be an adequate approximation when the increment step size is small with respect to the intrinsic period [39] [40]. The motor drivers are provided with a 5V PWM signal from the arduino outputs that rotate the ratchets to produce a single click or a "stridulation" of clicks which interestingly shares a similar kinetic mechanism with the plectrum of a cricket body [1].

5.3 Composition

Since rich complexity exhibited by collective systems is often a function of large-scale statistical processes, the installation was designed to bring out a dynamic range of collective behaviors. Given their sensitive nature of the parameters spaces, these systems are not conducive for meticulously planned approaches for deterministic outcomes. Rather, using the knowledge imparted from their dynamics, they encourage the careful design of constraints that will ultimately yield approximate results. The different processes associated with constructive accumulation were mainly used to evoke the presence of a seemingly autonomous and aware swarm of sound. Different outcomes are realized by modulating parameters associated with each of the collective behaviors.

5.3.1 Composition of Time points

The installation features different behavioral modes that are based on states associated with the coupled oscillator regimes above. These modes are written as routines in the code such that when they are triggered, the relevant parameters and time points are initialized and the generative model is allowed to play out. The routines themselves are randomly selected.

The parameters for each of the coupled oscillator algorithms were made time-varying by setting target values at discrete time points and then either stepping or linearly interpolated between values. Numerical simulation were performed in Python to approximate the output behavior of the system. For demonstration purposes, an example of six of the oscillators' time points in graph form is illustrated in Figure 4 which shows the time varying coupling coefficients (K_i) and the natural frequency dispersion, (σ_ω) overlayed with the center natural frequency, (ω_c). For this scenario with continuous Kuramoto oscillators, there is no coupling at the beginning with each oscillator simply moving at its natural frequency as dictated by natural frequencies chosen by the distribution with ω_c with variance σ_ω. The bottom Figure shows 5 different numerical simulations of the phase coherence magnitude, $|R(t)|$. This particular plot illustrates how sensitive the system is to initial conditions and changes in the frequency distribution. Time points were also created that allow oscillators to be enabled (turned on/off).

I provide a few descriptions of the ways in which each of the coupling processes are used in the piece and provide a timestamp in the documentation video (see footnote above) where this behavior occurs.

- Pulse-Coupling: Simple synchronization over different time periods as parameterized increasing coupling strength, ϵ, over time. Constructive accumulation tends to happen more progressively as compared to the other methods. (0:00 to 0:41).
- Scrambler oscillators: Synchronization with a relatively slow natural period (2:10 - 2:30).
- Kuramoto Oscillation: Synchronization as applied to rotational speed rather than pulses to produce a noisy timbral texture (00:52 - 01:09). Partial desynchronization after full synchrony (1:10 - 1:22). Partial synchronization by applying feedback to K_i using phase coherence state, R (1:30 - 2:10).

6. CONCLUSIONS

Ultimately, this artwork demonstrates one such artistic application of constructive accumulation as a generative process to explore different behaviors of synchronization as a sensory phenomena. Using ongoing research in coupled oscillation in dynamical systems modeling, this paper attempted to summarize these processes from both a perceptual and algorithmic perspective where sound onsets coalesce in time to create emergent rhythms. By presenting relevant research in auditory scene analysis and beat extraction/entrainment, I intended to suggest several approaches to the auditory and musical evaluation of such sensory phenomena and how they might encourage different listening modes and strategies. Future work would do well to look at more recent developments that may be particularly suitable for sonic engagement and interaction design in the audio-visual realm. One such area for such inquiry concerns Kuramoto models in higher dimensional spaces that have shown great promise for reproducing complex behaviors associated with swarms and flocking [25] [41].

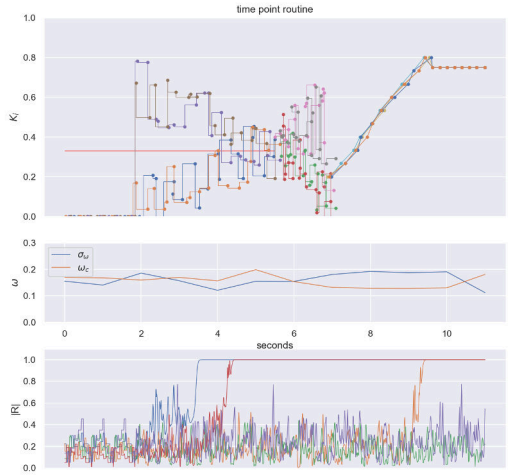


Figure 4. Timepoint plot for a single instance of a Kuramoto Routine. The top plot shows the coupling coefficients as a function of time for six of the nineteen oscillators. The red line indicates the critical coupling value. The middle plot shows the timepoints for the center frequency, ω_c, and the variance, σ_c, of the intrinsic frequencies of the oscillators. The bottom plot shows the phase coherence, R, over 5 numerical simulations. This demonstrates different behavioral outcomes that largely depend on the intrinsic frequency distribution.

7. REFERENCES

[1] M. Hartbauer and H. Römer, "Rhythm generation and rhythm perception in insects: The evolution of synchronous choruses," *Frontiers in Neuroscience*, vol. 10, no. MAY, pp. 1–15, 2016.

[2] B. B. H. Merker, "Synchronous Chorusing and Human Origins," *The Origins of Music*, no. 1999, pp. 315–327, 2000. [Online]. Available: http://www.biolinguagem.com/biolinguagem_antropologia/merker_2009_synchronouschorusing_humanorigins.pdf

[3] J. Buck, "Synchronous Fireflies," *Scientific American*, vol. 234, no. 5, pp. 74–85, 1976. [Online]. Available: https://www.jstor.org/stable/24968881

[4] S. C.-H. Yang, D. M. Wolpert, and M. Lengyel, "Theoretical perspectives on active sensing," *Current Opinion in Behavioral Sciences*, vol. 11, pp. 100–108, oct 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2352154616301255

[5] B. Morillon and S. Baillet, "Motor origin of temporal predictions in auditory attention," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 114, no. 42, pp. E8913–E8921, 2017.

[6] R. Barnes and M. R. Jones, "Expectancy, Attention, and Time," *Cognitive Psychology*, vol. 41, no. 3, pp. 254–311, 2000.

[7] M. R. Large, Edward W.; Jones, "Dynamics of Attending: How People Track Time-Varying Events," pp. 119–159, 1999.

[8] M. T. Elliott, A. M. Wing, and A. E. Welchman, "Moving in time: Bayesian causal inference explains movement coordination to auditory beats," *Proceedings of the Royal Society B: Biological Sciences*, vol. 281, no. 1786, 2014.

[9] U. Clayton, Martin; Sager, Rebecca; Will, "In time with the music: The concept of entrainment and its significance for ethnomusicology," *ESEM Counterpoint*, vol. 1, pp. 1–45, 2004.

[10] A. Neda, Z; Ravasz, E.; Vicsek, T.; Brechet, Y.; Barabasi, "Physics of the rhythmic applause," *Theoretical and Mathematical Physics*, vol. 104, no. 3, pp. 1104–1107, 1995.

[11] S. Chakraborty, S. Dutta, and J. Timoney, "The Cyborg Philharmonic: Synchronizing interactive musical performances between humans and machines," *Humanities and Social Sciences Communications*, vol. 8, no. 1, p. 74, dec 2021. [Online]. Available: http://www.nature.com/articles/s41599-021-00751-8

[12] A. Ravignani, D. Bowling, and W. T. Fitch, "Chorusing, synchrony and the evolutionary functions of rhythm," *Frontiers in Psychology*, vol. 5, no. SEP, pp. 1–15, 2014.

[13] J. Jalife, "Mutual entrainment and electrical coupling as mechanisms for synchronous firing of rabbit sino-atrial pace-maker cells." *The Journal of Physiology*, vol. 356, no. 1, pp. 221–243, nov 1984. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1113/jphysiol.1984.sp015461

[14] H. Wilson, "A critical review of menstrual synchrony research," *Psychoneuroendocrinology*, vol. 17, no. 6, pp. 565–591, nov 1992. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/030645309290016Z

[15] C. Peskin, *Mathematical Aspects of Heart Physiology*. New York: New York University, 1975.

[16] R. E. Mirollo and S. H. Strogatz, "Synchronization of Pulse-Coupled Biological Oscillators," *SIAM Journal on Applied Mathematics*, vol. 50, no. 6, pp. 1645–1662, dec 1990. [Online]. Available: http://epubs.siam.org/doi/10.1137/0150098

[17] J. J. Hopfield and A. V. Herz, "Rapid local synchronization of action potentials: toward computation with coupled integrate-and-fire neurons." *Proceedings of the National Academy of Sciences*, vol. 92, no. 15, pp. 6655–6662, jul 1995. [Online]. Available: http://www.pnas.org/cgi/doi/10.1073/pnas.92.15.6655

[18] A. I. Lavrova and V. K. Vanag, "Two pulse-coupled non-identical, frequency-different BZ oscillators with time delay," *Physical Chemistry Chemical Physics*, vol. 16, no. 14, pp. 6764–6772, 2014.

[19] K. P. O'Keeffe, P. L. Krapivsky, and S. Strogatz, "Synchronization as Aggregation: Cluster Kinetics of Pulse-Coupled Oscillators," *Physical Review Letters*, vol. 115, no. 6, 2015.

[20] N. Lem and Y. Orlarey, "Kuroscillator : A Max-MSP Object for Sound Synthesis using Coupled-Oscillator Networks," pp. 1–6.

[21] N. V. Lem, "Menagerie: Exploring the audio-visual rhythms of violence through data, gun triggers, and swarms," *Proceedings of the Sound and Music Computing Conferences*, vol. 2020-June, pp. 110–114, 2020.

[22] Y. Kuramoto, "Self-entrainment of a population of coupled non-linear oscillators," in *International Symposium on Mathematical Problems in Theoretical Physics*. Berlin/Heidelberg: Springer-Verlag, 1975, vol. 39, no. H, pp. 420–422. [Online]. Available: http://link.springer.com/10.1007/BFb0013365

[23] S. Petkoski and A. Stefanovska, "Kuramoto model with time-varying parameters," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 86, no. 4, pp. 1–7, 2012.

[24] R. Acebron, Juan; Bonilla, L.L, Vincente, Conrad; Ritort, Felix; Spigler, "The Kuramoto Model: a simple paradigm for synchronization phenomena," *Rev. Mod. Phys*, vol. 77, no. 1, pp. 137–185, 2005.

[25] X. Dai, X. Li, H. Guo, D. Jia, M. Perc, P. Manshour, Z. Wang, and S. Boccaletti, "Discontinuous Transitions and Rhythmic States in the D-Dimensional Kuramoto Model Induced by a Positive Feedback with the Global Order Parameter," *Physical Review Letters*, vol. 125, no. 19, pp. 1–6, 2020.

[26] T. Qiu, S. Boccaletti, I. Bonamassa, Y. Zou, J. Zhou, Z. Liu, and S. Guan, "Synchronization and Bellerophon states in conformist and contrarian oscillators," *Scientific Reports*, vol. 6, pp. 1–16, 2016. [Online]. Available: http://dx.doi.org/10.1038/srep36713

[27] N. Collins, "Errant sound synthesis," *Proceedings of International Computer Music*, 2008. [Online]. Available: http://classes.berklee.edu/mbierylo/ICMC08/defevent/papers/cr1049.pdf

[28] D. Pirrò, "COMPOSING INTERACTIONS," Ph.D. dissertation, University of Music and Performing Arts Graz, 2017.

[29] C. Roads, "Introduction to Granular Synthesis," *Computer Music Journal*, vol. 12, no. 2, p. 11, 1988. [Online]. Available: https://www.jstor.org/stable/3679937?origin=crossref

[30] D. Schwarz, "Corpus-based concatenative synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 92–104, 2007.

[31] C. Douglas, J. Noble, and S. McAdams, "Auditory Scene Analysis and the Perception of Sound Mass in Ligeti's Continuum," *Music Perception: An Interdisciplinary Journal*, vol. 33, no. 3, pp. 287–305, 2016.

[32] G. Essl, "Iterative Phase Functions on the Circle and Their Projections: Connecting Circle Maps, Waveshaping, and Phase Modulation," 2021, pp. 681–698. [Online]. Available: http://link.springer.com/10.1007/978-3-030-70210-6_44

[33] S. A. Shamma, M. Elhilali, and C. Micheyl, "Temporal coherence and attention in auditory scene analysis," *Trends in Neurosciences*, vol. 34, no. 3, pp. 114–123, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.tins.2010.11.002

[34] J. H. McDermott, M. Schemitsch, and E. P. Simoncelli, "Summary statistics in auditory perception," *Nature Neuroscience*, vol. 16, no. 4, pp. 493–498, 2013.

[35] J. H. McDermott and E. P. Simoncelli, "Sound texture perception via statistics of the auditory periphery: Evidence from sound synthesis," *Neuron*, vol. 71, no. 5, pp. 926–940, 2011.

[36] J. Simon and I. Winkler, "The role of temporal integration in auditory stream segregation." *Journal of Experimental Psychology: Human Perception and Performance*, vol. 44, no. 11, pp. 1683–1693, nov 2018. [Online]. Available: http://doi.apa.org/getdoi.cfm?doi=10.1037/xhp0000564

[37] B. C. J. Moore and H. Gockel, "Factors influences sequential stream segregation," *ACTA ACUSTICA UNITED WITH ACUSTICA*, no. August, 2001.

[38] N. V. Lem and T. Fujioka, "Extracting beat from a crowd of loosely coupled, concurrent periodic stimuli," *The Journal of the Acoustical Society of America*, vol. 146, no. 4, pp. 3049–3049, oct 2019. [Online]. Available: http://asa.scitation.org/doi/10.1121/1.5137568

[39] P. Östborn, "Phase transition to frequency entrainment in a long chain of pulse-coupled oscillators," *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, vol. 66, no. 1, pp. 1–9, 2002.

[40] T. Böhle, C. Kuehn, and M. Thalhammer, "On the reliable and efficient numerical integration of the Kuramoto model and related dynamical systems on graphs," *International Journal of Computer Mathematics*, 2021. [Online]. Available: https://doi.org/10.1080/00207160.2021.1952997

[41] K. P. O'Keeffe, H. Hong, and S. Strogatz, "Oscillators that sync and swarm," *Nature Communications*, vol. 8, no. 1, pp. 1–12, 2017. [Online]. Available: http://dx.doi.org/10.1038/s41467-017-01190-3

SMC-22 Paper Session 4

Algebraic Framework for Design and Qualitative Evaluation of Interactive Musical Systems Temporality

M.Desainte-Catherine, I.Durand
Univ. Bordeaux, CNRS, Bordeaux INP,
LaBRI, UMR 5800, F-33400 Talence, France
myriam@labri.fr

J.Haury, B.P.Serpette
SCRIME, Inria, Bordeaux
jeanhaury@gmail.com
Bernard.Serpette@inria.fr

S.Schwer
USPN, CNRS, LIPN UMR 7030
F-93430 Villetaneuse, France
schwer@lipn.fr

ABSTRACT

In this paper, we propose an algebraic framework for organizing interactive musical systems in temporal space in a way that helps in their design. The objective is to structure this space in order to be able to represent the existing and to open the exploration of new configurations that could inspire new musical practices. To do so, we propose some properties of interactive music systems allowing to compare them in a temporal lattice. This framework should be extended to other musical dimensions and give rise to optimizations of musical systems according to the desired performative properties.

1. INTRODUCTION

Until today, many studies and experiments have been carried out to create and use interactive musical systems (IMS) and many issues were discussed concerning their design on the software and hardware level as well as concerning their use. This paper proposes a study quite different from the previous ones because it introduces a theoretical framework for studying the temporal aspects at the design level and their implication on the uses, within an extension of the formal languages framework, which is an original algebraic approach compared to generalized intervals systems like [1] or geometry/topology of [2]. This algebraic study focuses on the temporal aspects of interactive musical systems and aims at describing and comparing different systems according to properties related to expressiveness and playability. In order to do so, we formalize interactive musical systems according to their temporal dimension only, forgetting any spatial information. That means that the system knows neither the musical information (such as pitches, durations, timbre and volume) nor the sensors nor even their number. One of the perspectives of this work is of course to extend it to the spatial dimension. For this purpose, this model should be confronted and articulated with the framework proposed in [3].

Thus, an interactive musical system temporality is represented by a tuple of algebraic objects which are the static and interactive scores, the interpretation space, the device language and the mapping function associating the musician's playing with the unfolding of the score. These elements allow to describe representation spaces, and to measure some properties of the systems. In particular, we propose a lattice of interactive scores that allows to organize this space according to a partial order. Finally, we introduce the notion of playing modes which corresponds to certain configurations that are distinguished from the musical point of view. Many playings modes can be defined with this formalism, whether they are existing or not.

The content of this article is limited to the presentation of this formalism and its interest to describe playing modes and measure some properties. The longer term goal is to automatically compute from a score interactive musical systems according to the desired playing modes, which depend on given constraints, such as the performer's motricity or the available devices.

We first present the notation that are useful to formalize the considered objects, then we present the elements that constitute a temporal interactive musical system and finally we show how to express playing modes with this formalism and how to measure properties. We present the rhythmic mode by using one or more keys to show how this study can help in the design of a system. The case of Jean Haury's Metapiano [4] has strongly inspired this work and is the subject of a particular development in this formalism.

2. PRELIMINARIES

In this section we present the definitions and notation that are used to formalize interactive musical systems. S-words [5] are used to model the temporal relations between the beginnings (ON) and the ends (OFF) of the notes of the score.

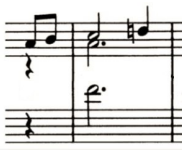



Figure 1. The first two bars of a Scherzo of Beethoven without any articulation

Let us consider the first two bars of the score of Beethoven's scherzo in the figure 1. The score is composed of five notes that provides our basic alphabet $\mathcal{A} = \{A4, B4, F4, C5, D5\}$ Each note in the score is associated with a sequence of playing intervals. An interval has two limits, an ON limit and an OFF one. Each is represented by an occurrence of a letter denoting the note. In order to discriminate ON and OFF, the occurrences denoting OFF are overlined, and the rank of each interval in the sequence is also provided with the corresponding occurrences. Thus,

Copyright: © 2022 M.Desainte-Catherine, I.Durand et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

any letter of $\mathcal{A}$ will occur twice as many times as the number of intervalles in the sequence. Half the occurrences - the odd ones are for the beginnings, the other half - the even one are for the ends and are overlined. In the Figure 1, the qualitative individual performance of A4 is modeled by the word : $[A4,1][\overline{A4},1][A4,2][\overline{A4},2]$, or, if their is no ambiguity, by $A4\overline{A4}A4\overline{A4}$; the word $B4\overline{B4}$ describes the one for B4. The traditional formal languages theory [6] is not able to take into account synchronization of events. S-languages does. A s-letter is a non-empty subset of $\mathcal{A}$. For instance, the s-letter $(\overline{A4}B4)$ [1] denotes that the end of the first play of A4 coincides with the beginning of B4.

This formalism allows to represent the following three cases of musical articulations :

- *Staccato* : $(A4)(\overline{A4})(B4)(\overline{B4})$
- *Legato* : $(A4)(B4)(\overline{A4})(\overline{B4})$
- *Esatto legato* : $(A4)(B4\overline{A4})(\overline{B4})$

The score in the figure 1 reduced to *esatto legato* articulations can be represented by the following s-word :
$sb = (A4)(\overline{A4}B4)(\overline{B4}F4A4C5)(\overline{C5}D5)(\overline{D5}F4\overline{A4})$.

The s-language associated to the score describes all the possible articulations, it is denoted $\mathcal{S}(\mathcal{A})$.

3. MUSICAL PIECE

This study is based on traditional scores of notes as they contain a large quantity of events linked by numerous temporal relations that the interpreter must take into account during the performance which eventually requires virtuosity. These scores also have a large repertoire. But this study is also adapted to any style of written music or music on a support, the notes being able to be more general sound objects. Thus a piece of electroacoustic music segmented in sound objects could be represented by a s-word.

A musical piece consists of a score and a space of interpretation, that is, a set of modifications of the piece that are allowed by the composer. In this paper a score is represented by a s-word augmented with musical information. Each interpretation of a score is also represented by a s-word so that a space of interpretation is represented by a set of s-words. The letters of the score could be associated with spatial musical information, such as pitches, durations and velocities, or any other information allowing to produce the sound result of the score.

In this study, we separate the question of temporality from that of the agogic interpretation. The first one is defined in the score, while the second one is defined in the space of interpretation below.

3.1 Interpretation

An interpretation of a score is a s-word, derived from the one of the score following rules that the musician must respect. In this article, the musician is not allowed to add or remove any event, that is, the s-language of interpretations $\mathcal{I}(sb)$ is included in $\mathcal{S}(\mathcal{A})$. For example, $\mathcal{I}(sb)$ may be defined by the following algebraic formula where we omit the indices of the occurrences of the intervals :
$\mathcal{I}(sb) = \{A4\}\{\overline{A4},\ B4\}\{\overline{B4},\ F4,\ A4,\ C5\}\{\overline{C5},\ D5\}\{\overline{D5},\ \overline{F4},\ \overline{A4}\}$
where $\{a,b\}$ means that a and b are not ordered [2]. In this article, we consider that the musician cannot interpret the simultaneity of events by pressing two keys exactly at the same time. Thus, in this case, $\mathcal{I}(sb)$ is simply a language and the number of its words, that is, the number of interpretations is : $|\mathcal{I}(sb)| = 1!2!4!2!3! = 576$. However, if she or he could apply *esatto legato* and synchronism of chords, $\mathcal{I}(sb)$ would be a set of s-words, and the number of interpretations would be $|\mathcal{I}(sb)| = 1 \times 3 \times 75 \times 3 \times 13 = 8775$.

3.2 Distance Between Interpretations

The distance between two interpretations i_1 and i_2 is defined according to the cost of operations applied to i_1 and leading to i_2. In a musical context, it is convenient to consider that the costs of these operations increase from the first to the fifth ones in the enumeration below.

Operations on interpretations.
Let $opi =< out, in, per_e, per_{es}, per_s >$ where the operations apply to adjacent s-letters :

- out : output of a letter out of a s-letter into a single s-letter
- in : input of a single s-letter inside an adjacent s-letter
- per_e : permutation of two end events
- per_{es} : permutation of an end event with a start event
- per_s : permutation of two start events

Transformation of interpretations. Let i_1 and i_2 be two interpretations with the same alphabet, there exists at least one $r = o(r_1, r_2, ..., r_n)$ being a composition of operations in opi such that $i_2 = r(i_1)$. Let us notice that in general r is not unique.

Cost of transformations. Let us associate a cost to each operation in opi and let us denote this cost by the function $cost$. Let i_1 and i_2 be two interpretations with the same alphabet, and let $r = o(r_1, r_2, ..., r_n)$ be a composition of operations in opi such that $i_2 = r(i_1)$, the cost of r is defined to be the sum of the costs of all the operations that constitute r : $cost(r) = \sum_i cost(r_i)$

Distance between two interpretations. Let i_1 and i_2 be two interpretations with the same alphabet, let O be the set of all r such that $i_2 = r(i_1)$, then the distance between i_1 and i_2 is defined to be : $\Delta(i_1, i_2) = min_{r \in O}(cost(r))$.

3.3 Degree of Freedom

Let s be a score and let $\mathcal{I}(s)$ be the space of interpretation of s. The size of $\mathcal{I}(s)$ is related to the degree of freedom given to the musician by the composer. Indeed, the smaller set of interpretations is limited to the score s, and the largest admits any kind of transformation. Let us denote by $\mathcal{A}*(s)$ the latter. Between the two of them, there exists a lot of possibilities (for example the one given for sb in subsection 3.1). Let us define the following quantity to measure the degree of freedom : $df(s, \mathcal{I}(s)) = \frac{|\mathcal{I}(s)|}{|\mathcal{A}*(s)|}$

3.4 Lattice of Interpretations

With the operations presented in the preceding subsection, it is possible to build a lattice [7] starting from the score and reaching all interpretations by applying all possible operations. Therefore, an arrow from i_1 to i_2 would mean that i_2 can be obtained from i_1 by applying one of the operations on i_1.

[1] We use parenthesis without comma instead of curly bracket because s-letters model leibniz'notion of instant, an equivalence classe for simultaneousness between instantaneous events. We use $\{[\overline{A4},1],[B4,1]\}$ to mean no order between the two occurences, which covers the three cases of articulations.

[2] We can get ab or ba or a and b strictly synchronized

4. INTERACTIVE SCORE

Let s be a score, an interactive score of s is a coloring of the s-word s, i.e. a pair (s, i) where i is a s-word included in s [3]. Thus, for a given score, there are several interactive scores that can be defined, depending on the associated s-word that is included. The smallest is the empty word noted ϵ and the largest is the one which is equal to the score s.

4.1 Interactive Points

Given an interactive score (s, i), the s-word i is called the *interaction s-word*. The letters of i are called *interaction points*. These letters will later be associated to an interaction device that will allow to activate them. Thus, the musician will be able to decide the precise moments of triggering these events. This will allow her or him to control the rhythmic and agogic expressiveness of the musical piece. Moreover, it should be remembered that the interactive points that are positioned in the same s-letter correspond to synchronized events on the score. These interaction points will allow the musician to activate these events in any order of her or his choice, in particular to desynchronize attacks, or chose how to articulate successions of notes.

Example. Let (sb, i) an interactive score where sb is the s-word of the score in figure 1. We can represent the interactive score by coloring in red the interaction points within the s-word of the score : $sb = ([A4, 1])([\overline{A4}, 1][B4, 1])([\overline{B4}, 1][F4, 1][A4, 2][C5, 1])([\overline{C5}, 1][D5, 1])([\overline{D5}, 1][\overline{F4}, 1][\overline{A4}, 2])$. So, the interaction s-word is the following : $i = ([A4, 1])([B4, 1])([\overline{C5}, 1][D5, 1])$. This interactive score has 4 interaction points that are the 4 letters of i : $[A4, 1], [B4, 1][\overline{C5}, 1][D5, 1]$. Finally, we see that according to the s-word indices, we keep the correspondence between the letters of i and s : $i[1] = ([A4, 1]) \subset p[1], i[2] = ([B4, 1]) \subset p[2], i[3] = ([\overline{C5}, 1][D5, 1]) \subset p[4]$.

4.2 Degree of Interactivity

The degree of interactivity of an interactive score $is = (s, i)$ is equal to the number of letters of the s-word i, that is, the number of interaction points, divided by the total number of letters of the score : $di(is) = \frac{|i|}{|s|}$ Thus, a null degree of interactivity indicates that there is no interaction point and that it is therefore impossible to control the progress of the score, while a degree of interactivity equal to 1 indicates that all the events of the score, whether they are beginnings or ends of notes, are activated by the musician.

4.3 Partial Order of Interactive Scores

Let $is_1 = (s, i_1)$ and $is_2 = (s, i_2)$, we say that is_1 is less interactive than is_2 if and only if i_1 is included in i_2. $is_1 < is_2 \iff i_1 \subset i_2$. Let us notice that if $is_1 < is_2$ then $di(pi_1) \leq di(pi_2)$.

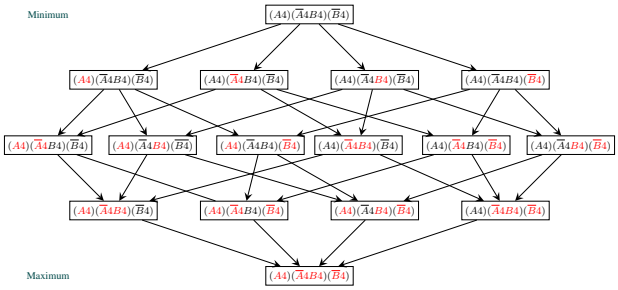
Thus, for a given s, there is only one interactive score of minimal order which is (s, ϵ), where ϵ denotes the empty s-word. There is no possible interaction with this interactive score. The next higher order interactive scores are those which admit one interaction point. There are as many of them as there are letters in s, since this interaction point can be located on any of these letters.

[3] i is said to be included in s if it is obtained by deleting letters of s.

Finally, there is only one maximum order interactive score which is (s, s), because its s-word of interaction contains all the letters of s. So all possible s-words of interaction for the score s are included in it, by definition. This order is indeed partial and not total because some elements are not comparable.

4.4 Example of Lattice

Let us consider the following score : $s = (A4)(\overline{A}4B4)(\overline{B}4)$. It has 4 possible points of interaction. There are 15 interaction s-words that can be associated with this score. The interactive scores can be organised in a lattice according to the partial order introduced in the preceding paragraph, where $is1 \longrightarrow is2$ means that $is1 < is2$. In the following representation of the lattice, the minimum is located at the top and the maximum at the bottom. We have omitted the indexes of the letters in order to save space.



5. TEMPORAL INTERACTIVE MUSICAL SYSTEM

In this section, we define all the elements that constitute interactive musical systems and introduce several quantities to study their properties. This definition is limited to the temporal aspect of the interactive musical systems. This is why they are called temporal interactive musical systems (TIMS).

5.1 Definition of a TIMS

A temporal interactive musical system is a tuple $< is, \mathcal{I}(s), \mathcal{D}, m >$, where :

- s is a score
- $is = (s, i)$ is a colored s-word representing an interactive score
- $\mathcal{I}(s)$ is the space of interpretation of s
- $\mathcal{D}$ is the language of the device
- m is a command mapping that associates an interpretation of s to a word of $\mathcal{D}$

where $\mathcal{D}$ and m are defined just below.

5.2 Device Language

In order to interact with an interactive score, it is necessary to define the physical device that allows the activation of the interaction points. For discrete controls, we can use sensors such as buttons or keyboard keys. Depending on the type of interaction desired, several keys can be hold down simultaneously.

Two actions are possible on a key, the press and the release. Any sequence of actions on a key corresponds to a sequence of presses followed by releases.

To describe the language of a key, we associate a letter x, an occurrence of x for pressing the key, and an occurrence of $\overline{x}$ for releasing the key. One cannot press twice the same key without releasing it before the second time. Thus, the key's language is described with the following rational expression : $T^1_x = (x\overline{x})^*$

If the sensor has n keys, let x_1, x_2, ... x_n, $\overline{x}_1$, $\overline{x}_2$, ... $\overline{x}_n$ the letters corresponding to the presses and releases of these keys, and let $T^1_{x_i}$ be the language of the key x_i, then the language of this sensor is defined by the shuffle product [4] (denoted by $\sqcup\!\sqcup$) of the languages $T^1_{x_i}$ of each of these keys :

$$T^n_{x_1,\dots,x_n} = \sqcup\!\sqcup_{i=1}^{n} T^1_{x_i}$$

In this article, we are interested in languages corresponding to the temporal dimension. To limit this study to this dimension, we consider that the system does not know the keys of the device and does not even know how many there are. Thus, all keys are combined into one generic key. In this case, the press or the release of a key is associated with an event in the score, via the s-word of interaction. This event does not depend on the key, but only on the position of the press or release in the interaction s-word.

Thus, let us denote $T^1_{x^n}$ the projection language of a n-key sensor in which all keys are represented by the same letter x. Thus the press of any key is denoted by x and any release by $\bar{x}$. In others words, we consider the trace of the presses and the releases. The only constraint is that the ith occurrence of $\bar{x}$ must occur after the ith occurrence of x.

This language is the set of Dyck words [5] of maximum height n, that is, the set of well-parenthesized words with a maximum of n embeddings of parenthesis. However, for these words, a closing parenthesis is always associated with the last free opening parenthesis (which has not yet been associated with a closing parenthesis). It may therefore be musically useful to study other kind of languages. Indeed, in the physical play of the keyboard, a release is rather associated with the press of the key that is released. However, it should be noticed that the number of words may differ according to the different associations of keypresses and releases that are considered. Let us notice in particular the following Dyck's word of $T^1_{x^2}$: $w = xx\overline{x}x\overline{x}\overline{x}$, there exist 4 words of $T^2_{x_1,x_2}$ that correspond to it :

$w_1 = x_1x_2\overline{x}_1x_1\overline{x}_1\overline{x}_2$; $w_2 = x_1x_2\overline{x}_1x_1\overline{x}_2\overline{x}_1$;
$w_3 = x_1x_2\overline{x}_2x_2\overline{x}_1\overline{x}_2$; $w_4 = x_1x_2\overline{x}_2x_2\overline{x}_2\overline{x}_1$;

Because we assume that the first non overlined letter that is read is x_1, the second one x_2 and so on. Let us notice here that the word w_2 corresponds to the choice of the command mapping used in the Metapiano that is presented in 6.4.2.

In the examples of the next section, the languages of the device $\mathcal{D}$ that will be considered are chosen among the languages of this subsection.

5.3 Command Mapping

Let s be a score, $is = (s, i)$ an interactive score and $< is, \mathcal{I}(s), \mathcal{D}, m >$ be a TIMS. Let us first define a *command* as a colored word of $\mathcal{D}$ and let $\mathcal{C}$ be the set of commands, that is the set of all colorings of the words of $\mathcal{D}$.

$$\mathcal{C} = \{(w, wi), w \in \mathcal{D} | wi \subset w\}$$

where w is a word in the language of the device $\mathcal{D}$ and wi is called the *command word*.

A *command mapping* is a function associating an interpretation in $\mathcal{I}(s)$ to a command in $\mathcal{C}$:

$$m : \mathcal{C} \longrightarrow \mathcal{I}(s)$$

Let us point out that the words w and wi are sequences of letters corresponding to presses or relesases of keys. Now, each letter of the command word wi is mapped to a letter of the interaction s-word i. Thus each letter in wi allows to activate the corresponding interaction point in i which itself allows to trigger the corresponding event in the score s. Let us notice that many choices have to be made in the design of the function m.

- In the definition of the mapping between wi and i, the kind of the letter of wi, that is press or release, is not necessarily related to the kind of the event in the score, that is start or end of a note. Indeed, a press may activate an end of note and vice-versa. However, for reasons of musical ergonomics, the types of the letters should match each other most of the time.
- To match the letters of wi with those of i, chronological order is preferred for reasons of real-time play. But in some cases, i can have s-letters including several interaction points. Then it is necessary to decide how to choose the interaction point of i depending on the letter of wi that the musician has just played.

Let us define the following sets :

- Let $\mathcal{C}(m)$ be the domain of m, $\mathcal{C}(m) \subset \mathcal{C}$
- Let $\mathcal{I}(m)$ be the codomain of m, $\mathcal{I}(m) \subset \mathcal{I}(s)$

Example. Let $< is, \mathcal{I}(s), T^1_x, m >$ be a TIMS. Let us consider the following interactive score :

$$is = (A4)(\overline{A4}B4)(\overline{B4})$$

The interactive s-word of is is $i = (A4)(B4)(\overline{B4})$
Let us define a mapping m reduced to one command c, so that the set $\mathcal{C}$ is defined as follows : $\mathcal{C}(m) = \{c\}; c = x\overline{x}x\overline{x}$
The command word of c is the word wi defined as follows :

$$wi = xx\overline{x}$$

The mapping m associates the red letters of is with the red letters of c in a chronological way. More precisely, m makes the following associations :

- the first x of wi is associated with the interactive point of the first s-letter of i, that is the event A4. Thus the first press of a key activates the beginning of the note $A4$.
- the second x of wi is associated with the interactive point of the second s-letter of i, that is the event B4. Thus the second press activates the beginning of the note $B4$, as well as the release of the note $A4$, because those two events are synchronised in the same s-letter of is.
- Finally the second release is associated with the interactive point of the third s-letter, that is the event $\overline{B4}$. Thus the second release activates the end of the note $B4$.
- The resulting interpretation is exactly the score s with the durations that the musician has played.

The first release $\bar{x}$ of the command c is inactive, but it is necessary so that the word c belongs to the language of the device $\mathcal{D}$ which guarantees that this word can actually be played by the musician on this device.

[4] The shuffle product of languages $L_1, L_2, ..., L_n$ is the set of words obtained by interleaving in all possible ways the words of $L_1, L_2, ...L_n$

[5] The number of Dyck word of size k is called the Catalan number and is denoted by $\mathcal{C}_{k/2}$.

5.4 Expressiveness and Difficulty of a TIMS

Let s be a score, and let $ms =< is, \mathcal{I}(s), \mathcal{D}, m >$ be a temporal interactive musical system, let n be the size of the command words in $\mathcal{C}(m)$ and let $\mathcal{D}^n$ be the set of the words of $\mathcal{D}$ of size n : $\mathcal{D}^n = \{w \in \mathcal{D}, |w| = n\}$. Let us define the following quantities associated to a TIMS :

- Degree of difficulty : $dd(ms) = \frac{|\mathcal{D}^n)| - |\mathcal{C}(m)|}{|\mathcal{D}^n|}$
- Degree of expressiveness : $de(ms) = \frac{|\mathcal{I}(m)|}{|\mathcal{I}(s)|}$

5.5 Improving a TIMS

According to a given objective, there are several ways to improve a TIMS by changing the definitions of its elements.

- To increase expressiveness one can increase $\mathcal{I}(m)$, eventually as well as $\mathcal{I}(s)$
- To decrease difficulty, increase $\mathcal{C}(m)$ or reduce $\mathcal{D}$
- The decrease of $\mathcal{D}$ may imply the decrease of $\mathcal{C}$, thus may imply the decrease of $\mathcal{I}(m)$. So that a balance must be found between difficulty and expressiveness.

6. PLAYING MODES OF TIMS

In this section, we show several playing modes that can be defined for TIMS in order to illustrate the formalism and show how to compare them according to the properties that were introduced before.

6.1 The Score

In all the examples below, we use the score which is displayed in figure 1. There are many ways to represent a score by a s-word. Let us recall the s-word sb :
$(A4)(\overline{A4}B4)(\overline{B4}F4A4C5)(\overline{C5}D5)(\overline{D5}F4A4)$.
If we project this s-word on the onset letters, we obtain the successions of the beginnings of the notes and the synchronizations. We see that all the successions are represented and that the synchronizations are exact :
$(A4)(B4)(F4A4C5)(D5)$.
In particular, every succession is represented by *esatto legato*, whether the notes are actually linked or not, like in this example : $(A4)(\overline{A4}B4)(\overline{B4})$

6.2 Rhythmic Mode

This mode allows the musician to play with the rhythm using one or more keys. Let us first present the TIMS with one key and second the TIMS with two keys in order to compare them.

We want to define a TIMS allowing to play sb with a sequence of eight steps of key presses and releases. Let us define our TIMS as the following :

$$ms_1 =< is_1, \mathcal{I}_1(sb), \mathcal{D}_1, m_1 >$$

where the elements $is_1, \mathcal{I}_1(sb), \mathcal{D}_1$ and m_1 are defined below.

6.2.1 *Space of Interpretation*

Let us now define the space of interpretation $\mathcal{I}_1(sb)$ that we wish to allow the musician to play. This mode provides only one interpretation in term of s-word, because all the temporal relations of the score have to be strictly respected.

$$\mathcal{I}_1(sb) = \{sb\}$$

6.2.2 *Interactive Score*

In this case, the interactive score has exactly one interactive point per s-letter. As a consequence, the choice of the letter carrying the interactive point does not matter in practice. However, in this example, a maximum of start letters in the first voice have been selected for musical ergonomic reasons. The interactive score is defined as follows, the interactive points displayed in red, every line representing a bar and omitting the indices for the sake of clarity of the notation.

$is_1 = (sb, i_1) = (A4)(\overline{A4}B4)$
$(\overline{B4}F4A4C5)(\overline{C5}D5)$
$(\overline{D5}F4A4)$

6.2.3 *One Key Device*

Let us consider that the device is reduced to one key. Thus, the alphabet is $\{a, \bar{a}\}$ and $\mathcal{D}_1 = T^1_a = (a\bar{a})^*$

6.2.4 *Command Mapping*

The command mapping is defined by the set of valid commands $\mathcal{C}(m_1)$, i.e. the set of commands leading to an interpretation in $\mathcal{I}(sb)$. It is defined as a set of colored word (w, wi) where w is in $\mathcal{D}_1$ and $wi \subseteq w$, the subword wi being represented by the subword colored in red below. In this example, we have only one command : $\mathcal{C}(m_1) = \{a\bar{a}a\bar{a}a\bar{a}a\bar{a}\}$
The command word wi is mapped to the interaction s-word i_1 of the interactive score is_1. Thus the three black letters are inactive, their purpose is to immerse the command into the language of the device $\mathcal{D}_1$.
Here is how the mapping works :
$a \longrightarrow A4$, $\bar{a} \longrightarrow \emptyset$, $a \longrightarrow \overline{A4}B4$, $\bar{a} \longrightarrow \emptyset$, $a \longrightarrow \overline{B4}F4A4C5$, $\bar{a} \longrightarrow \emptyset$, $a \longrightarrow \overline{C5}D5$, $\bar{a} \longrightarrow \overline{D5}F4A4$

The command mapping can be defined as follows :
$\forall c \in \mathcal{D}_1, m_1(c) = \perp \; if \; c \notin \mathcal{C}(m_1) \; else \; m_1(w) = sb$
Thus we have :

- $\mathcal{C}(m_1) = \{c\}, n = |c| = 8$
- $\mathcal{I}(m_1) = \{sb\}$

6.2.5 *Properties*

This TIMS allows the musician to play a piece of music with total respect for the temporal relationships with the *esatto legato* that are imposed for the first four notes. The musician cannot make mistakes so that the difficulty is minimum. Indeed, the reduction of the device to one key leads to only one command so that the degree of difficulty is minimum. The total respect of the relations implies that there is only one possible interpretation so that the expressiveness is at its maximum, while the degree of freedom is low.

Let us precise the quantities that are useful to measure the properties of our TIMS. The size of the commands is $n = 8$, the number of words of the device of size 8 is $|\mathcal{D}_1|_8 = 1$, the number of interpretations is $|\mathcal{I}_1(sb)| = 1$, and the number of all possible permutations of the score is $|\mathcal{A}*(sb)| = D(4, 2, 2, 2, 2)$ [6] with $D(4, 2, 2, 2, 2) >> D(2, 2, 2, 2, 2) = 2244361$. $|\mathcal{C}(m_1)| = 1, |\mathcal{I}(m_1)| = 1$.

- Degree of freedom : $df(s, \mathcal{I}_1(sb)) = \frac{|\mathcal{I}_1(sb)|}{|\mathcal{A}*(sb))|} < 4.45e^{-07}$

[6] This is a generalized Delannoy number, whose computation is given by the following recursive formula [see for instance (Durand-Schwer 2008) : Let $(p_1, p_2, \cdots, p_n)$ n natural integers, and $\mathcal{P}red(p_1, \ldots, p_n)$ = $\{(\widetilde{p_1}, \cdots, \widetilde{p_n}) \in \mathbb{N}^n | \; if \; p_i \neq 0$, then $\widetilde{p_i} \in \{p_i, p_i - 1\}$ else

- Degree of interactivity : $di(is_1) = |i_1|/|s| = 5/12 = 0.416$
- Degree of difficulty : $dd(ms_1) = \frac{|\mathcal{D}_1|_8 - |\mathcal{C}(m_1)|}{|\mathcal{D}_1|_8} = \frac{1-1}{1} = 0$
- Degree of expressiveness : $de(ms_1) = \frac{|\mathcal{I}(m_1)|}{|\mathcal{I}_1(sb)|} = 1/1 = 1$

6.2.6 *Device With More Than One Key*

Let us consider now that the device has two keys and let us consider that those two keys are generic. Thus, the language of our device is the projection of the language of two keys : $\mathcal{D}_2 = T^1_{x^2}$

This is the set of Dyck words of maximum height 2. Let us consider the following TIMS that differs from ms_1 only in the number of keys : $ms_2 = (s, is_1, \mathcal{I}_1(sb), \mathcal{D}_2, m_1\}$

This system has the same properties as ms_1 except that the degree of difficulty is much higher. Indeed, the difficulty increases with the number of keys because of the increasing number of possible words of the device. The reason for that is that a lot of words of the device are wrong. In our example, the number of words of length 8 of the device is 8 and only one of those words is correct. So the musician has a $7/8$ chance of being wrong. In particular, all the words beginning by two successive a, that is by two successive key presses, are wrong. For example, the following word is wrong because of the second letter that we have colored in red : $c' = a\bar{a}\bar{a}a\bar{a}a\bar{a}$

In this example, only one word is correct and all other possible words the musician can play with the device are wrong. The situation gets worse if the number of keys increases. Moreover we omitted to count the series of presses and releases that would leave keys pressed at the end. This further increases the number of wrong words.

There are several ways to reduce the difficulty in order to improve this TIMS (see subsection 5.5). For example one may increase the number of right words by changing the mapping. The actual mapping associates only one word with the interaction word of the interactive piece, this word being a series of key presses and releases. We propose the two following ways to modify the mapping :

- One possibility is not to consider the nature of the action (press or release) and to associate any serie of actions to the interaction word. Thus, the word c' becomes correct with this mapping, and produces the same result as c.
- Another possibility is to reject every wrong letter. Indeed, since at any time, the system can predict the right letter, it may reject all letters that are different from the right one. The inconvenient of this choice is that the musician could think that the system is not working well because it does not always react to her or his actions. Thus, with this mapping, the word c' is the beginning of a correct word. Since the second letter a is rejected, the second letter $\bar{a}$ has to be rejected also, so that the word has to be extended by 2 correct letters, or even more if there are more upcoming wrong letters. For example, the following command is correct with this new mapping : $a\bar{a}\bar{a}a\bar{a}a\bar{a}a\bar{a}$

$\widetilde{p_i} = p_i,\ 1 \leq i \leq n\} - \{(p_1, \cdots, p_n)\}$.

$$D(p_1, \ldots, p_n) = \sum_{\mathcal{P}red(p_1,\ldots,p_n)} D_n(\widetilde{p_1}, \cdots, \widetilde{p_n})$$

with $D_1(0) = 1$, $D_n(p_1, \ldots, p_{n-1}, 0) = D_{n-1}(p_1, \ldots, p_{n-1})$. For instance, if all $p_i = 2$, then, the sequence $(D(p_1, \cdots, p_n))_n$ is the sequence A055203 of the Sloane's on-line encyclopedia of integer sequences.

6.3 The Metapiano

Another way to reduce the difficulty of the previous TIMS is to increase its space of interpretation as well as its set of valid commands. This is the choice made in the *Metapiano* [7] of Jean Haury [4]. Like the temporal interactive musical systems we consider in this paper, the Metapiano has no timing information (rhythm and duration) and cannot play automatically. In fact, in the score of the Metapiano, are only associated to the s-letters note pitches and their belonging to a particular musical voice of polyphony; no other temporal (duration) or dynamic (velocity) information. However, many interpretations can be performed on a single or multiple sensors, or keyboard key.

Let us define the Metapiano by the following tuple :

$$mtp = \{is_2, \mathcal{I}_2(sb), \mathcal{D}_9, m_2\}$$

Only the score is the same as the previous examples, all the other elements change. Indeed, the space of interpretation is open to some permutations of beginning and ending notes in order to allow to play *legato* or *staccato* successive notes. Therefore, the interactive score admits more interactive points in order to allow those permutations, and the mapping has to be adapted. And $\mathcal{D}_9$ is defined below.

6.3.1 *Space of Interpretation*

The space of interpretation $\mathcal{I}_2(sb)$ of the Metapiano is a set of s-words made from the s-word of the score. In this space, the beginnings of notes are ordered between them as well as the ends of notes for all the s-words of $\mathcal{I}_2(sb)$. Moreover the end of a note always comes after its beginning. To do so, the beginnings and endings of notes that are in a same s-letter of the s-word s are desynchronised. More precisely, the s-letters of s containing ending and beginning letters are splitted into two s-letters, one containing the endings, and the other one containing the beginnings.

Let us project the s-word of the score sb on the two kinds of letters, as we did in the subsection 6.1, and consider the two sequences of s-letters, on the left are the s-letters corresponding to a projection on the onset letters, and on the right to the projection on the offset letters :

$a_1 = ([A4, 1])$
$a_2 = ([B4, 1])$
$a_3 = ([F4, 1][A4, 2][C5, 1])$
$a_4 = ([D5, 1])$

$\bar{a}_1 = ([\overline{A4}, 1])$
$\bar{a}_2 = ([\overline{B4}, 1])$
$\bar{a}_3 = ([\overline{C5}, 1])$
$\bar{a}_4 = ([\overline{D5}, 1][\overline{F4}, 1][\overline{A4}, 2])$

Now the a_i are considered as letters in an alphabet and we consider words without synchronization.
The temporal constraints between those letters in the space of interpretation are the following :

- For $1 \leq i \leq j \leq 4$, the s-letter a_i comes before the s-letter a_j
- For $1 \leq i \leq j \leq 4$, the s-letter $\bar{a}_i$ comes before the s-letter $\bar{a}_j$
- All the s-letters containing an *ON* letter x comes before the s-letter containing the *OFF* letter $\bar{x}$.

The temporal relations can be represented graphically by the lattice which is displayed on figure 2, where the arrows denote a temporal precedence relationship.

[7] see for example the small fugue of Förster: https://youtu.be/FdMWvQpQnPM or Beethoven Great Fugue Op. 133 https://youtu.be/0hDlWxA6SlY

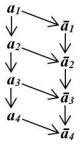


Figure 2. The lattice representing $\mathcal{I}_2(sb)$

In particluar, this lattice indicates that a_2 and $\bar{a}_1$ may be performed in any order but not synchronized, and both after a_1 and before $\bar{a}_2$.

The language corresponding the lattice is the set of the 14 possible articulations between the notes of the score :

$$\mathcal{I}_2(sb) = \{a_1\bar{a}_1a_2\bar{a}_2a_3\bar{a}_3a_4\bar{a}_4, a_1\bar{a}_1a_2a_3\bar{a}_2\bar{a}_3a_4\bar{a}_4, a_1a_2\bar{a}_1\bar{a}_2a_3\bar{a}_3a_4\bar{a}_4, a_1a_2\bar{a}_1a_3\bar{a}_2\bar{a}_3a_4\bar{a}_4, a_1a_2a_3\bar{a}_1\bar{a}_2\bar{a}_3a_4\bar{a}_4, a_1\bar{a}_1a_2a_3\bar{a}_2a_4\bar{a}_3\bar{a}_4, a_1\bar{a}_1a_2\bar{a}_2a_3a_4\bar{a}_3\bar{a}_4, a_1a_2\bar{a}_1\bar{a}_2a_3a_4\bar{a}_3\bar{a}_4, a_1a_2\bar{a}_1a_3\bar{a}_2a_4\bar{a}_3\bar{a}_4, a_1a_2a_3\bar{a}_1\bar{a}_2a_4\bar{a}_3\bar{a}_4, a_1\bar{a}_1a_2a_3a_4\bar{a}_2\bar{a}_3\bar{a}_4, a_1a_2\bar{a}_1a_3a_4\bar{a}_2\bar{a}_3\bar{a}_4, a_1a_2a_3\bar{a}_1a_4\bar{a}_2\bar{a}_3\bar{a}_4, a_1a_2a_3a_4\bar{a}_1\bar{a}_2\bar{a}_3\bar{a}_4\}$$

Figure 3 illustrates the temporal relations between three successive notes according to this space of interpretation. In this figure, for x and y being a letter a_i or $\overline{a_i}$ with $1 \leq i \leq 3$, an arrow going from a letter x to a letter y denotes that y is played just after x.

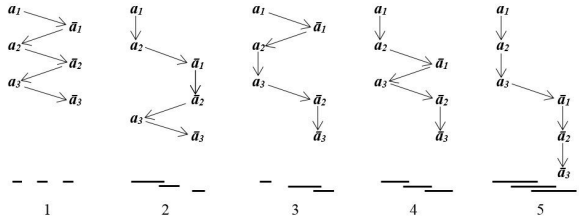


Figure 3. The five possible articulations of three notes and the qualitative temporal relations between the three associated intervals : the leftmost one for a_1, the middle one for a_2 and the last one for a_3

6.4 Interactive Score

There are many possibilities to define the interactive score. Let us consider the following one which is quite close to the previous examples.

$$is_2 = (s, i_2) = (A4)(\overline{A4}B4)(\overline{B4}F4A4C5)(\overline{C5}D5)(\overline{D5}F4\overline{A4})$$

In this interactive score, three interaction points have been added compared to the previous examples so that the endings of the three first notes are interactive. The interaction s-word i_2 has 8 letters. This way, the parity between the beginnings and the ends of notes in i_2 is restored and the mapping will be simplified.

6.4.1 Device of the Metapiano

The Metapiano is usually played on a 9-key keyboard whose keys are generic. This number has been chosen for reasons of musical ergonomics, especially for fast playing. Let us base the study with such a device keys. Thus, the language of our device is the projection of the language of nine keys :

$$\mathcal{D}_9 = T^1_{x^9}$$

6.4.2 Command Mapping

In this mapping all the letters are active, so that the command word is always equal to the whole command. In other words, all the letters are red. The set of valid commands $\mathcal{C}(m_2)$ is the set of all the words of length 8 that can be obtained from the device with a succession of presses and releases of any key and when all the keys are relaxed at the end. However, the words that do not have as much a as $\overline{a}$ are not in this set. This set can be obtained from the word $(a\overline{a})^4$ by swapping the a letters with the $\overline{a}$ letters to bring them to the left or simply from $\mathcal{I}_2(sb)$ by erasing the indices.

$$\mathcal{C}(m_2) = \{ a\bar{a}a\bar{a}a\bar{a}a\bar{a}, a\bar{a}aa\bar{a}\bar{a}a\bar{a}, a\bar{a}\bar{a}a\bar{a}a\bar{a}\bar{a}, a\bar{a}\bar{a}\bar{a}a\bar{a}a\bar{a}, aa\bar{a}a\bar{a}a\bar{a}\bar{a}, aa\bar{a}\bar{a}a\bar{a}\bar{a}a\bar{a}, aa\bar{a}\bar{a}\bar{a}a\bar{a}a\bar{a}, aaa\bar{a}\bar{a}\bar{a}a\bar{a}\bar{a}, aaa\bar{a}\bar{a}a\bar{a}\bar{a}\bar{a}, aa\bar{a}aa\bar{a}\bar{a}\bar{a}\bar{a}, aaa\bar{a}a\bar{a}\bar{a}\bar{a}, aaaa\bar{a}\bar{a}\bar{a}\bar{a}\}$$

The mapping between a command word and the interaction s-word proceeds as follows.

- the ith letter a of the command word corresponds to the ith beginning letter of the interaction s-word.
- the ith letter $\overline{a}$ of the command word corresponds to the ith ending letter of the interaction s-word.

Thus, the word c' that was considered wrong in the example 6.2.6 becomes correct with this mapping. This word is the third one of the set of commands $\mathcal{C}(m_2)$ above. Here is how the mapping works with this word :
$a \longrightarrow A4, a \longrightarrow B4, \overline{a} \longrightarrow \overline{A4}, \overline{a} \longrightarrow \overline{B4}, a \longrightarrow C5F4A4, \overline{a} \longrightarrow \overline{C5}$,
$a \longrightarrow D5, \overline{a} \longrightarrow \overline{D5F4A4}$

We notice that the permutations of the letters in the command word imply permutations of the corresponding s-letters of the score via the interaction s-word. Since the device does not allow to release a key before it has been pressed, the temporal constraints between the a and $\overline{a}$ letters are always respected. Moreover the genericity of the keyboard keys implies that permutations between the letters a are not possible since they are not distinguishable. It is the same for the letters $\overline{a}$. Thus, all the words of the set $\mathcal{D}_9$ of size 8 are in $\mathcal{C}(m_2)$ and vice-versa.

The interpretation resulting from the execution of this command word is the following : $(A4)(B4)(\overline{A4})(\overline{B4})(C5F4A4)$ $(\overline{C5})(D5)(\overline{D5F4A4})$. Let us examine the distance of this interpretation to the score sb. We can see that the out operation must be applied three times to the score $sb2$ in order to obtain this interpretation. From left to right, the first application was made on the letter $B4$, the second one on the letter $\overline{B4}$ and the last one on the letter $C5$. Thus, the distance of this interpretation to the score is equal to three times the cost of the out operation.

6.4.3 Properties

This TIMS allows the musician to play a piece of music with respect for some temporal relationships between the events and with the possibility of swapping some events of the score. The mapping is such that its domain is equal to the set of words of length 8 of the device language, that is the 4th number of Catalan. The number of words that the musician can play on the device is much higher, because she or he can play prefixes of Dyck words of longer size. For example she or he can do more presses than the number of beginnings of notes of the score. So the musician can play some wrong words. At last, in this example, the codomain is equal to the interpretation space.

Let us precise the quantities that are useful to measure the properties of our TIMS. Let us denote by $|\mathcal{D}_9|_{8+}$ the number of words of size 8 that are prefixes of longer Dyck words. This number is much higher than $|\mathcal{D}_9|_8$. $n = 8$, $|\mathcal{D}_2|_{8+} \geq \mathcal{C}_4$, $|\mathcal{I}_2(sb)| = \mathcal{C}_4$, $|\mathcal{A}*(sb))| > 2244361$, $|\mathcal{C}(m_2)| = \mathcal{C}_4$, $|\mathcal{I}(m_2)| = \mathcal{C}_4$, where $\mathcal{C}_4$ is the 4th number of Catalan, that is 14.

- Degree of freedom : $df(s, \mathcal{I}_2(sb)) = \frac{|\mathcal{I}_2(sb)|}{|\mathcal{A}*(sb))|} <$ 6.23E-06
- Degree of interactivity : $da(is_1) = |i_2|/|s| = 8/12 = 0.66$
- Degree of difficulty : $dd(ims_1) = \frac{|\mathcal{D}_9|_{8+} - |\mathcal{C}(m_2)|}{|\mathcal{D}_9|_{8+}} > 0$
- Degree of expressiveness : $de(ims_1) = \frac{|\mathcal{I}(m_2)|}{|\mathcal{I}_2(sb)|} = 1$

It can be seen that in the two cases of TIMS considered, the rhythmic mode and the Metapiano, the degree of expressiveness is at its maximum. They both take advantage of the device to allow the musician to explore all the space of interpetation allowed by the composer. The metapiano offers a larger space of interpretation, which is expressed by the greater degree of freedom than the rhythmic mode. The Metapiano requires more gestures from the musician, which is expressed by a higher degree of interactivity. Finally it is a little more difficult to play than the rhythmic mode because the musician must anticipate the end of the score to stop all the notes in time.

7. CONCLUSION

In this paper we have presented an algebraic framework to describe the temporal aspects of interactive musical systems. For this purpose we have limited our study to systems using generic discrete control sensors. Indeed, we have eliminated all spatial knowledge concerning the musical information but also the sensor information. Thus, only the information concerning the temporal relations between the static events of the score and the real time events of the performance are considered. This study allows to dissociate the temporal constraints coming from the written score, from those coming from the interpretation domain opened by the composer to the musicians, and those coming from the sensors. The formal definition of these spaces opens the possibility to study them, to browse them, to measure them, and to compare various solutions for designing temporal aspects of interactive musical systems. Some examples are provided in this article, but many others could be studied and new system models could be identified.

This study opens the way to several new perspectives. One of them is to complete the study of playing modes that show musical interest. Distinguished playing modes could be formalized and integrated into interactive sequencers such as OSSIA score [8, 9]. Some questions arise, for example, how to simplify the playing of a score for a musician with a motor disability while ensuring a satisfactory degree of expressiveness? This requires detecting moments of interest in the score to place the points of interaction so that the interpretation is fluid and expressive. For this purpose, it could be useful to focus on models of expressivity like [10].

The extension to the space domain of this model can be done in several steps. First of all, a recognition of the device keys and the notes of the score will allow to know at any time which key is released and consequently which note is to be finish. This step may allow to model the second version of the Metapiano, which gave rise to the development of the Midifile Performer [11]. The study in [12] presents also various playing modes that are of interest.

Finally, a major objective is to be able to optimize TIMS according to certain criteria, in particular playability and difficulty, in order to find the best solutions in the algebraic spaces defined here according to the objectives of the design of interactive musical systems.

Acknowledgments

This research was hosted by the SCRIME of the University of Bordeaux (Studio of Creation and Research in Computer Science and Experimental Music) and funded by the french ministry of culture. The authors would like to thank wormly Gyorgy Kurtag Jr for bringing them together, and for helping them with his deep vision in artistic research.

8. REFERENCES

[1] T. M. Fiore, T. Noll, and R. Satyendra, "Morphisms of generalized interval systems and pr-groups," *Journal of Mathematics and Music*, pp. 3–27, 2013.

[2] G. Mazzola, S. Göler, and S. Müller, *The Topos of Music, Geometric logic of Concepts, Theory and Performance*. Birkhaus Verlag, 2005.

[3] J. Malloch, D. Birnbaum, E. Sinyor, and M. Wanderlay, "Towards a new conceptual framewok for digital musical instruments," in *DAFX'06*, 2006, pp. 49–52.

[4] J. Haury, "Un répertoire pour un clavier de deux touches. théorie, notation et application musicale," *Document numérique*, vol. 11, pp. 127–148, 2008. [Online]. Available: https://www.cairn.info/revue-document-numerique-2008-3-page-127.html

[5] I. Durand and S. Schwer, "A tool for reasoning about qualitative temporal information: the theory of s-languages with a lisp implementation," *J. of Universal Computer Science*, vol. 14 (20), pp. 3282–3306, 2008.

[6] S. Ginsburg, *The Mathematical Theory of Context Free Languages*. McGraw-Hill Book Company, 1966.

[7] G. Birkhoff, "Lattice theory. american mathematical society," *Providence, RI*, 1967.

[8] M. Desainte-Catherine, A. Allombert, and G. Assayag, "Towards a hybrid temporal paradigm for musical composition and performance: The case of musical interpretation," *Computer Music Journal*, vol. 37, no. 2, pp. 61–72, 2013.

[9] J.-M. Celerier and al, "Ossia: towards a unified interface for scoring time and interaction," in *First International Conference on Technologies for Music Notation and Representation, Paris, France*, 2015, pp. 81–90.

[10] C. Eduardo, Cancino-Chacón, T. Gadermaier, G. Widmer, and M. Grachten, "An evaluation of linear and non-linear models of expressive dynamics in classical piano and symphonicmusic," *Mach Learn, Springer-Verlag Nature*, pp. 887—909, 2017.

[11] J. Chabassier, M. Desainte-Catherine, J. Haury, M. Pobel, and B. P. Serpette, "Midifileperformer: a case study for chronologies," in *FARM*, 2021, pp. 13–22.

[12] T. Lucas, S. D. Laubier, and C. D'Alessandro, "Animer la musique sur support," in *JIM 2020. Université de Strasbourg/Faculté des Arts*, 2020.

JS Fake Chorales: a Synthetic Dataset of Polyphonic Music with Human Annotation

Omar Peracha
Humtap
omar@humtap.com

ABSTRACT

High-quality datasets for learning-based modelling of polyphonic symbolic music remain less readily-accessible at scale than in other domains, such as language modelling or image classification. Deep learning algorithms show great potential for enabling the widespread use of interactive music generation technology in consumer applications, but the lack of large-scale datasets remains a bottleneck for the development of algorithms that can consistently generate high-quality outputs. We propose that models with narrow expertise can serve as a source of high-quality scalable synthetic data, and open-source the JS Fake Chorales, a dataset of 500 pieces generated by a new learning-based algorithm, provided in MIDI form.

We take consecutive outputs from the algorithm and avoid cherry-picking in order to validate the potential to further scale this dataset on-demand. We conduct an online experiment for human evaluation, designed to be as fair to the listener as possible, and find that respondents were on average only 7% better than random guessing at distinguishing JS Fake Chorales from real chorales composed by JS Bach. Furthermore, we make anonymised data collected from experiments available along with the MIDI samples. Finally, we conduct ablation studies to demonstrate the effectiveness of using the synthetic pieces for research in polyphonic music modelling, and find that we can improve on state-of-the-art validation set loss for the canonical JSB Chorales dataset, using a known algorithm, by simply augmenting the training set with the JS Fake Chorales.

1. INTRODUCTION

Representation learning for application to music has become an increasingly active field of research in recent years, as Figure 1 demonstrates. Despite growth in the domain, certain bottlenecks have persisted which hinder the rate of breakthroughs, particularly regarding problems related to symbolic music. The most obvious of these concerns the volume of high-quality datasets available for researchers to work with cheaply and with low effort; datasets comprising millions of samples [1] or even tens of billions of canonical input/output pairs [2] have been available for many years in the fields of computer vision and natural language processing. Some of these are even licensed with sufficient permissiveness as to allow commercial use of any technology whose development relied upon said data [3],

Copyright: © 2022 Omar Peracha et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

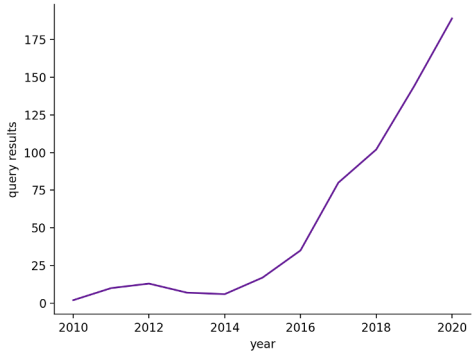


Figure 1. Number of results returned by arXiv search when filtering for papers in the Computer Science, Mathematics or Statistics categories with both "Music" and "Learning" in their abstracts for the years 2010-2020 inclusive. For years before 2015, we further filter out any irrelevant results manually.

a factor which plays an important role in determining the ultimate viability of deploying these algorithms into the real world where they may generate value for businesses and customers alike, naturally promoting further investment. Unsurprisingly, the aforementioned two domains continue to see incredible progress in research and increasingly wide adoption of the resulting technology in consumer-facing products.

A second bottleneck more pertinent to generative music modelling is the lack of automated methods of assessing outputs which strongly correspond to good qualitative performance. This difficulty in measuring success naturally leads to slower progress. Few alternatives have been proposed to human evaluation, which is typically slow or costly, for assessing the quality of algorithmically-generated music. The process of acquiring human feedback itself tends to vary in implementation details between different published works [4–6], which further convolutes the process of reliably comparing results in the literature.

We propose a dataset of synthetic polyphonic music, generated by our neural network-based *KS_Chorus* algorithm, in symbolic form, and aim to validate both its usefulness for aiding in symbolic music modelling tasks and its ability to scale on demand. We achieve the latter by building the dataset from 500 consecutive generated outputs of *KS_Chorus*, with zero cherry-picking of samples, and performing an experiment designed to solicit human evaluation in a context entirely free from curation, which may otherwise serve to skew results in the algorithm's favour. 500

samples is small enough that conducting such an experiment remains fairly manageable, but is large enough relative to the set of 382 chorales by Johann Sebastian Bach [7], whose style the JS Fake Chorales seek to imitate, that the results may be deemed significant. After 6,810 responses collected from an estimated 446 unique respondents, our experiment shows that human listeners are only 7.3% better than a coin flip at telling apart our synthesised samples from chorales composed by Bach himself. We facilitate scalability by making a web app freely available for researchers to generate new samples from *KS_Chorus*, [1] enabling them to grow the dataset, which is further described in Section 3.3.

We then conduct experiments using the JS Fake Chorales as training data or auxiliary data on a common benchmark task, namely modelling the canonical JS Bach Chorales dataset. Using the current state-of-the-art model, *TonicNet* [8], without any tuning of parameters, we show that training purely on JS Fake Chorales achieves performance close to state-of-the-art on the JS Bach Chorales in terms of validation set loss, even without training on any pieces from the actual JS Bach training set. By combining the JS Bach Chorales training set with the JS Fake Chorales, using the latter as a form of dataset augmentation, we can improve on state-of-the-art results. We make all code to run our ablation studies publicly available, for reproducibility. [2]

In evidencing the scalability of the JS Fake Chorales and their usefulness in common music modelling tasks, we hope to highlight the possibility for synthetic data to help alleviate the current bottleneck of low data volume in this domain. Finally, to help address the bottleneck of weak methods for measuring qualitative performance, we make anonymised metadata for each of the 6,810 human responses available as part of the dataset, [3] which includes information about the evaluator's level of music education and which specific pieces were guessed correctly or incorrectly by which evaluator, among other data further detailed in Section 3. Future work may see attempts at modelling these data as a means to automatically infer qualities which make music more readily-perceived as human-composed.

Ours is the first dataset in this domain that provides such data for such a large number of samples without cherry-picking, to our knowledge. Similarly, Section 4.1 shows that *KS_Chorus* is the first deep learning algorithm proven to consistently approach, though not yet achieve, a production-level consistency of high-quality unconditioned outputs in Bach's, or indeed any clearly-defined polyphonic, style. The value of this is that the generated samples can already be very useful as auxiliary training data in downstream tasks, as shown in Section 4.2.

In a data-starved domain, models with narrower focus are able to squeeze performance on specific genres from the lack of a requirement to generalise to other styles, for example by making style-specific adjustments in the representation. As a concrete example, *KS_Chorus* does not need to handle individual voices capable of producing more than one note simultaneously, allowing for a more efficient representation of Bach's chorales than cross-genre polyphonic music models, such as *MuseNet* [9]. We propose that narrow-focussed models, by generating high quality synthetic data in a single or small number of styles, may play a role in significantly increasing the availability of training data, thus empowering future generative algorithms to generalise better across styles while improving on both quality and consistency.

[1] Web app to generate new samples is available at `https://omarperacha.github.io/make-js-fake/`

[2] Code to reproduce results from experiments is available at `https://github.com/omarperacha/TonicNet`

[3] Dataset and usage information are available at `https://github.com/omarperacha/js-fakes`

2. RELATED WORK

Synthetic datasets in the music domain have so far seen wider use in audio contexts rather than symbolic music. For example, both [10] and [11] trained algorithms on synthetic data to learn how to accurately detect the fundamental frequency of monophonic musical material. While algorithms which process audio have seen more benefit from synthetic data up until now, examples do exist of such data playing a role in symbolic music contexts; in [12], extra training data is synthesised by performing a conditional nonlinear transform on the original training corpus, in a naive 1-dimensional token embedding space. This data is used to augment the training set and is fed to the generator and discriminator in a generative adversarial learning context, with an input label corresponding to the intensity of the transform performed on the original data sample so as to allow both networks to learn features expressed in the extra data which are also present in the original data. It is worth noting that the intent in this case was not to create perfect stylistic imitations of the original dataset, but to introduce novelties in perceivably-systematic ways. Nonetheless, [12] serves as a precursor for using synthetic data to improve the quality of generated symbolic music.

With 10,854 complete pieces of solo piano music in MIDI form, [13] is among the larger symbolic music datasets available. It could be considered a synthetic dataset in that the authors used an automated method of transcription [14] to convert over 1000 hours of audio recordings into a symbolic representation. The authors report validation onset and offset F1 score of 0.825, and a score of 0.967 for onsets alone. These are impressive results in the current landscape, yet clearly this may translate to many transcription errors across a dataset comprising millions of note events. The severity of these inconsistencies with respect to the stylistic integrity of the original human-composed piece can be hard to measure, and further work is required to determine the areas where this dataset will ultimately enable progress in the field.

Both [4] and [5] propose algorithms intended to generate stylistic imitations of Bach's chorales, and use human evaluation to support claims of the effectiveness of their solutions. The experiment conducted by [4] first takes 50 pieces from the validation set of the Bach chorales and uses their *DeepBach* algorithm to reharmonise them. They repeat the process with other baseline algorithms. They then take 12-second extracts of each and play these samples one-by-one, asking listeners to determine if they think each is by Bach or is a generated piece. In contrast, we generate 500 pieces entirely from scratch with no conditioning and always play the entire piece to listeners, which can be over one minute in length. We also make a ground truth chorale by Bach available to the listener to play back at any time alongside the sample currently being evaluated.

The experiment conducted by [5] is similar to ours in that it allows to listen to two samples simultaneously. Unlike ours, however, the listener must simply pick which of the

two samples they feel is more Bach-like. In our case, the evaluator knows that one sample is definitely composed by Bach, and must decide if the second sample is also by Bach or if it is generated by *KS_Chorus*. A second significant difference is that [5] only use a set of 12 compositions fully-generated by their *BachBot* algorithm in their experiments. This is too small a sample to substantiate the model's general performance. In our case, we generate 500 samples in a row and include all of these in the human evaluation experiment without cherry-picking. Neither [4] nor [5] make a substantial dataset of generated compositions available, while we make these initial 500 available as the JS Fake Chorales dataset, and offer evidence that the dataset can be further scaled while maintaining the same quality consistently as a result of avoiding curation during human evaluation.

Perhaps the most relevant work to ours is the Bach Doodle Dataset [15], obtained by allowing users to interact with the *CoCoNet* algorithm [6]; the user could enter a melody via a web interface, and the algorithm would harmonise this melody in the style of Bach. Comprising 21.6 million samples, the size far exceeds any that is commonly-used in symbolic music, to our knowledge. The web interface also collects a user rating on a three-point scale to reflect their satisfaction with the output. This rating can not be said to reflect stylistic consistency with any intended target, however, but simply provides a general measure of the user's experience. This rating is provided as part of the dataset, along with more metadata about the user's session. Because the melodies are user-entered, only constrained to fit within a selected key and within a 2-bar duration, the quality of the final samples in the Bach Doodle Dataset will be inherently unpredictable. Furthermore, all samples are fixed to 2 bars in duration, and melodies are limited to quaver resolution. Our samples consist only of pieces in their entirety, with a median bar duration of 11.375 and maximum of 35.5, and rhythmic resolution can fully capture that seen in the original Bach chorales.

Several style-agnostic polyphonic models have been proposed in recent years. Three examples notable for being particularly high-performing are *MuseNet* [9], *MMM* [16] and *MusicBERT* [17]. While *MMM* and *MusicBERT* are highly promising, *MuseNet* is the only one of these to be evaluated on full song generation. This evaluation is not performed quantitatively, and provided samples are cherry-picked, though an interface for novel song generation is available. It is likely that all three models have been trained on the Bach Chorales, though not known for certain as not all the datasets are public. In any case, none of these models are able to demonstrate the capability to generate a dataset with quality matching the JS Fake Chorales.

3. DATASET

3.1 Synthesising 4-part Chorales

To create the JS Fake Chorales dataset, we first develop a learning-based sequence modelling algorithm, *KS_Chorus*, and train it on the Bach chorales as made available by the music21 toolkit [18]. Bach's chorales are a good choice as the basis for a synthetic dataset mainly because they are widely-cited in the literature regarding algorithmic modelling of polyphonic music [4–8], and so downstream quantitative analysis against good benchmarks is readily possible.

KS_Chorus is a generative algorithm for polyphonic music of any number of instruments, where each instrument is itself monophonic. It consists of an ensemble of deep RNN-based networks with some domain-specific architectural additions, totalling roughly 130 million parameters, and a sampling routine idiosyncratic both to this architecture and the data representation utilised. We do not detail the specifics of the architecture, input representation or training process here, and instead leave this to a separate work for the future.

Once trained, we then sample 500 compositions from the model consecutively and fully unconditionally, i.e using absolutely no priming or similar input to specify parameters such as the length of the compositions. The only exception to this relates to metre; while a small handful of pieces exist in the original training corpus which are not in 4/4 time, we found during development that *KS_Chorus* falls somewhat below par when generating compositions in other time signatures. We therefore manually restrict all generated samples to be in 4/4, because the likelihood of being correctly identified as algorithmically-composed seemed to be consistently higher in other time signatures.

Each generated composition is compared to each of the 344 Bach chorales in the training set seen by the model during development to ensure originality; in particular, the edit distance to every single sample in the training corpus must be greater than 50%, measured separately across both the entire piece and across the opening few bars. An edit distance below 50% in either case would invalidate the sample and trigger the algorithm to run once more. Furthermore, all samples in the original dataset were transposed as far as possible in either direction while maintaining a singable range for each voice, and the edit distance was similarly validated to these transpositions. A single composition takes roughly 10 minutes on average to generate when running on CPU. The algorithm was implemented and trained using the PyTorch library [19], and sampling was also performed in Python.

While we did not measure the edit distance of generated pieces to all other generated pieces at sample time, we perform post-hoc analysis of this metric. Since we do not detail the representation used by *KS_Chorus* in this work, for the sake of reproducibility we perform these comparisons using a standard representation seen in the literature [8, 20]; the piece is split into 16th-note time-steps and represented as a matrix $x \in \mathbb{Z}^{4\times T}$, where T is the number of time-steps in that sample, the four rows represent the four different voice parts and the value of each element is the pitch observed in the respective voice at the respective time-step. This matrix is typically converted into a linear sequence before being input to a learning algorithm, by end-concatenating the columns. We also make the JS Fake Chorales available in this representation.

Using the above representation, we compare the intra-set edit distance (i.e. comparing each sample with every sample from the same dataset) for both the Bach chorales and the JS Fake Chorales, and we repeat the comparison of each JS Fake chorale with each Bach chorale. Table 1 shows the results of taking the smallest edit distance from each sample in dataset 1 to each sample in dataset 2 for the three different dataset configurations, and computing the mean of these smallest distances. We also provide the minimum in each case. We can see that there is in fact more intra-set similarity

Dataset 1	Dataset 2	Minimum	Mean
JSF	JSB	55.078%	72.818%
JSF	JSF	25.625%	66.657%
JSB	JSB	12.755%	57.792%

Table 1. Results of taking the smallest edit distance from each sample in dataset 1 to each sample in dataset 2, and computing the mean and the minimum of these values.

among the Bach chorales than among the JS Fakes. Of course, the edit distance between the two was enforced at generation time, hence the distribution of inter-set least edit distances trending much higher than that of either intra-set least edit distances. We can deduce from this post-hoc analysis that 50% was perhaps an unnecessarily strict threshold for edit distance during the process of sampling from *KS_Chorus*.

3.2 Human Annotation

We conduct a human evaluation experiment hosted on the internet to investigate how these generated compositions are perceived in comparison to Bach's chorales. We first present the experiment design in this section and describe the data obtained and provided as part of the JS Fake Chorales dataset, while the actual results of this experiment will be discussed in Section 4. We use $\mathcal{G}$ to notate the set of 500 pieces which comprise the JS Fake Chorales dataset, and $\mathcal{C}$ to refer to the training corpus of 344 chorales by J.S. Bach.

3.2.1 Listening Experiment

Participants are first prompted to self-report their highest level of musical education, selecting from a set of six coarse pre-defined options: 0, no musical education; 1, some musical education, but no formal qualifications; 2, high School level qualifications directly related to music; 3, undergraduate degree directly related to music; 4, postgraduate degree directly related to music; and 5, postgraduate degree specialising in the music of J.S. Bach.

Once they have confirmed their choice, they are taken to a page with instructions describing how to complete the experiment. On this page, the participants are provided with a reference chorale by Bach to listen to, chosen randomly per session from the 344 pieces $[C_1, ..., C_{344}] \in \mathcal{C}$. Participants are informed that the given chorale is a genuine example of Bach's music, and they must play it through to the end in order to advance further. Upon doing so, a second piece will appear, which is selected at random from the combined set of all pieces $\mathcal{G} \cup \mathcal{C}$ (excluding whichever Bach chorale was used as the reference), with equal probability of any sample in this set being selected.

Participants are not informed whether this second piece is composed by Bach or *KS_Chorus*, but are tasked to determine this. Once more they can not proceed without listening to the given sample in its entirety, at which point a menu will appear through which they can provide their response and submit. All pieces are rendered using the same piano synthesiser, and can be played as many times as the participant wants before submitting a response, including the reference track. *KS_Chorus* does not currently include tempo data as part of the representation it learns to model, therefore all piece are played back at a fixed tempo of 120bpm.

Upon submitting a response, participants are taken to a new page where they are immediately shown the correct answer, and given the option to try another round. They are also shown a running score which increments by 1 for each correct response in the session, which carries over if they choose to try a new sample. Should they choose to do so, they will be taken back to the previous page where the same reference piece will be available for them to play back as desired, and a new sample will be loaded for the participant to identify as belonging to $\mathcal{G}$ or $\mathcal{C}$, ensuring this new sample has not previously been seen in the current session. In this case, participants are not forced to replay the reference track, but must once more listen through the new sample entirely before proceeding.

Responses were solicited both organically via social media, and through Amazon MTurk [21]. In the latter case, participants were required to obtain a score of 10 in order to fulfil the conditions for payment, and were given a 30 minute limit to do so. This was done to motivate participants to try their utmost to provide correct responses; mandating that the sample be played through entirely for each round vastly increases the time cost for every wrong answer, meaning that random guessing is not an effective strategy for MTurk respondents to earn the available reward. The mean time taken by MTurk workers to complete the task was 19m45s, and we estimate these individuals make up roughly 66% of all unique respondents. Overall we received 6,810 responses from 446 unique IP address hashes. Responses were collected throughout the month of February 2021, and the website remains live for anyone to interact with, [4] while data is no longer being collected from sessions.

3.2.2 Metadata

Besides a simple binary value denoting the accuracy of a participant's response, we collect several other data points designed to offer insight on how complex a given sample was to identify by the participant. Examples include how many times the sample was played before submitting a response and how long the participant took to submit once they had heard the sample for the first time. We aggregate these and make them available as part of the dataset in the hope that this information might provide value to researchers.

For each generated chorale $G_i \in \mathcal{G}$, we make four additional pieces of metadata available: (1) Total responses concerning sample G_i; (2) responses which correctly identified G_i as composed by an algorithm; (3) mean number of times the sample was played before a response was submitted; and (4) mean time in seconds between hearing the sample and submitting a response. We provide these data separately for each skill level, to provide further granularity on how the samples were perceived by different demographics. This allows such custom analysis as examining which pieces were likely to be misidentified by people with university degrees in music, and which pieces did people with limited musical education find simplest to correctly identify. We also provide the data aggregated over all skill levels. Further usage explanation is provided in the documentation.

[4] Listening test available at `https://omarperacha.github.io/listening-test/`

Skill	**Bach**			**A.I.**			**Total Samples**
	Samples	Correct	Ratio	Samples	Correct	Ratio	
0	705	339	48.1%	1081	633	58.6%	1786
1	943	445	47.2%	1351	748	55.4%	2294
2	515	270	52.4%	732	409	55.9%	1247
3	264	98	37.1%	366	225	61.5%	630
4	215	97	45.1%	348	223	64.1%	563
5	120	65	54.2%	170	83	48.8%	290
ALL	2762	1314	47.6%	4048	2321	57.3%	6810

Table 2. Number of samples by Bach and by *KS_Chorus* seen over all participant sessions during human evaluation (columns 2 & 5), the number of times these were correctly identified (columns 3 & 6) and the resulting rate of correct responses (columns 4 & 7), shown across all skill levels.

3.3 Web Application for Generating New JS Fake Chorales

We make a web application freely available for anyone to sample new chorales with *KS_Chorus*. The application consists of a single page, including text and a button to trigger sample generation. Once the sampling process is complete, the button is replaced with a MIDI player where the user can preview the newly-generated sample, rendered with a piano synthesiser. The MIDI file will also be automatically downloaded to the user's local hard drive from their browser at that time. The main JS Fake Chorales repository will be periodically updated to include all samples generated from the web app, and users are informed of this. No user data of any kind is captured from this interface, besides the generated sample itself. While we commit to the persistency of the JS Fake Chorales dataset, and all samples generated from this web app in future included therein, we may in due course choose to discontinue the availability of this app as we see fit, for example if the cost of upkeep is deemed no longer sustainable.

4. EXPERIMENTAL RESULTS

4.1 Human Evaluation

4.1.1 Sample Category Identification Task

We compare the likelihood of participants determining a sample to be human-composed, and present results across skill levels for samples by both Bach and *KS_Chorus*. Given an ideal model whose distribution of generated compositions $\widetilde{G}$ exactly equals the distribution of training samples $\widetilde{C}$, we would expect the likelihood of any $G_i \in \mathcal{G}$ and $C_i \in \mathcal{C}$ being deemed as human-composed to also be equal. Given a fair experiment which does not incur bias, for example by disproportionately highlighting certain samples of Bach's pieces or suppressing specific qualities present in $\mathcal{C}$ from being observed by the evaluators, this expected likelihood should be 50%.

Table 2 shows the proportion of all responses which correctly identified samples from $\mathcal{G}$ and $\mathcal{C}$, broken down by skill level. We see that the total rate of correct responses for $\mathcal{C}$ is 47.6%, while for $\mathcal{G}$ it is 57.3%. These values, though unequal, are close to each other, and in particular they suggest evaluators did not differ greatly from random chance regarding the likelihood of a correct guess for either corpus. This is despite receiving instant feedback after each round and a reference $C_i \in \mathcal{C}$ being made accessible at all times while prompting for a response, which should both aid in increasing the participants' success rate.

We perform Fisher's exact test on a contingency table consisting of rows $\mathcal{C}$, $\mathcal{G}$ and columns *voted Bach*, *voted A.I.* across all skill levels to interpret whether samples from $\mathcal{G}$ did indeed have a similar likelihood to those from $\mathcal{C}$ of being considered human-composed. We test each skill group separately, the combination of all skill groups together, and two subgroups dividing the respondents into those who self-reported holding a formal qualification in music, and those who did not. We also compute the F1 score for each group as a measure of overall performance on the classification task. We report these results in Table 3, and see that at the 99.9% critical value we can accept the null hypothesis that $\mathcal{C}$ and $\mathcal{G}$ are equally likely to be voted as being written by Bach, for every individual skill group and for the two larger subgroups. However, the value of p varies greatly across these groups, and for the overall sample comprising all responses $p = 6.432e^{-5}$, so we must reject the null hypothesis. Therefore while at a high level the JS Fakes are certainly difficult to distinguish from Bach Chorales, performing further tests makes it clear there is room for improvement in the quality of the generated samples. Closing this gap is precisely an example of an application which this dataset might serve in future.

We can also see from Table 3 that there is no apparent trend in F1 score across respondent skill levels, which seems surprising. One obvious candidate for this is the inherent noise and unreliability introduced by enabling respondents to self-report their skill. A stronger method for a future experiment would involve assessing user skill level via some short test. However, the fact that $p > 0.001$ consistently across all skill groups suggests participants within each group did in fact respond similarly to their peers. Another criticism we can make is that while likelihood of considering a piece from either sample as human-composed was similar, this observed value was not 50%, but 55.7% as measured by weighted average recall. While we have theories for why this may be the case, we must conclude that there are likely some aspects of the experimental design which are not free from introducing small bias. We leave the experiment fully accessible online for transparency and leave it for future work to analyse and improve on the fairness of designing such an experiment.

4.1.2 Ranking

One possibility offered by the data captured is the ability to rate samples with respect to the likelihood that a listener might perceive the pieces as human-composed. This could

SKILL	0	1	2	3	4	5	NO QUAL	QUAL	ALL
p	0.006	0.233	0.004	0.740	0.033	0.635	0.006	0.003	< 0.001
F1	0.609	0.576	0.590	0.594	0.647	0.540	0.591	0.600	0.594

Table 3. Results of performing Fisher's exact test and computing F1 score for each skill group, all skill groups combined, and two subgroups dividing the respondents into those who do and do not hold a formal qualification in music.

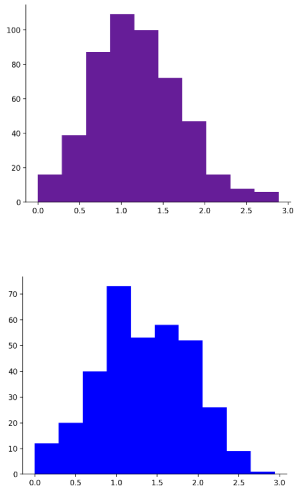


Figure 2. Histogram of scores R for pieces in the JS Fakes Dataset (top) and by Bach in the training dataset (bottom).

prove a useful feature to researchers whose goal is to understand the qualities of music which people tend to perceive as "human", or to implicitly model them as part of a generative system designed to output increasingly higher-quality symbolic music. Moreover, this can be done in a manner stratified by listener music education level, which may provide further insight into which qualities are preferred or more telling for which people. We use the following equation to obtain such a rating R for any given sample $S_i \in \mathcal{G} \cup \mathcal{C}$.

$$R_{S_i} = N_{S_i}^{-0.5} * (\beta_{S_i} + \frac{\epsilon}{N_{S_i}}) \quad (1)$$

Equation 1 is designed to balance the absolute ratio of responses deeming the sample to belong to $\mathcal{C}$ with the number of responses received for that particular sample in total, since this ratio becomes more reliably indicative of quality as the number of responses increases. We use β_{S_i} to denote the number of responses categorising a sample as Bach's music, and N_{S_i} for the total number of responses for that sample. We use 0.001 for ϵ, which serves to preserve a logical ranking for any samples where $\beta_{S_i} = 0$, by giving those which received more negative responses overall a lower score.

We plot R as histograms for both $\mathcal{G}$ and $\mathcal{C}$ in Figure 2. We can see that samples from $\mathcal{C}$ trend towards slightly higher scores than those from $\mathcal{G}$, suggesting that pieces by Bach are indeed still more likely to be perceived as human-composed than those by *KS_Chorus*. Having said that, the distributions of R for the two sets show strong similarities on aggregate. The maximum R for the generated pieces $\mathcal{G}$ is 2.89, while the mean and standard deviation are 1.19 and 0.55 respectively. For $\mathcal{C}$, the maximum, mean and standard deviation are 2.94, 1.33 and 0.57 respectively.

We perform a Mann-Whitney U test on the R scores for the two sets and find $p = 0.0493$, which supports our claims that there are similarities between the way the two sets are perceived. Furthermore, we perform the same test on a naive scoring method, for transparency, where the rating is given by the number of "Bach" votes for each sample divided by the total number of votes for that sample. In this case, $p = 0.303$. We believe this motivates the use of Equation 1 to calculate rating, as it implicitly factors in a form of confidence by multiplying by a term proportional to N_{S_i}, offering a more nuanced impression of listener perception.

4.2 Algorithmic Evaluation

We conduct experiments to ascertain the benefit of using the JS Fakes Dataset to help with music modelling tasks. We use the canonical JSB Chorales dataset to perform ablation studies, and measure performance in terms of negative log likelihood on the validation set, using the same dataset split as [7]. The task involves element-wise autoregressive modelling of chorales represented in sequence form, as described in Section 3.1. We use the *TonicNet_Z* algorithm to perform experiments, using the same hyperparameters, training schedule and specific data representation as [8]. We explore the effect of training only on the JS Fake Chorales and of combining them with the training set of the JSB Chorales as a form of data augmentation. We show how the JS Fake Chorales impact results both with and without augmentation by transposition, and also investigate the impact of limiting the set of JS Fake Chorales seen by the model to those samples with the highest R ratings. Table 4 shows the results of these experiments, alongside two further strong-performing algorithms.

In the first column of Table 4, we denote the combinations and subsets of datasets used to train *TonicNet_Z*. *Bach* here refers to the 229 samples in the JSB Chorales training set and JSF refers to the full set of 500 pieces in the JS Fake Chorales dataset. We also use two special subsets of *JSF* in experiments, namely *JSF-top* and *JSF-top-s*; these represent the subset of *JSF* obtained by ordering the samples by their R scores and taking the top 229. In the case of *JSF-top-s*, we first consider only responses by participants with a skill level of 3 or higher before calculating R, to perform an initial investigation into whether insights specifically from individuals with higher levels of musical education might impact downstream results.

The third column shows whether any augmentation techniques were performed on the datasets in question. *Trnsps* refers to transposing the training set only, as far as possible in both direction while all voices in the sample remain within a singable range for their respective voice type. *MM* refers to a crude technique of converting pieces in major keys to minor, and vice versa, as described in [8], roughly doubling the dataset size. The remaining columns show experimental results on the validation set, where *NCL* denotes

Datasets	**Model**	**Aug.**	**Val. NLL**	**NCL Val. NLL**
Bach†	*CoCoNet (ordered)*	None	-	0.436
Bach†	*Music Transformer*	None	-	0.335
JSF-top	*TonicNet_Z*	None	0.474	0.363
JSF	*TonicNet_Z*	None	0.424	0.319
Bach*	*TonicNet_Z*	None	0.422	-
JSF-top-s	*TonicNet_Z*	Trnsps	0.364	0.272
JSF-top	*TonicNet_Z*	Trnsps	0.364	0.269
JSF	*TonicNet_Z*	**Trnsps**	**0.328**	**0.234**
Bach*	*TonicNet_Z*	Trnsps	0.321	0.224
Bach*	*TonicNet_Z*	Trnsps+MM	0.317	0.220
Bach+JSF-top	*TonicNet_Z*	Trnsps	0.309	0.215
Bach+JSF-top-s	*TonicNet_Z*	Trnsps	0.307	0.213
Bach+JSF	*TonicNet_Z*	**Trnsps**	**0.300**	**0.208**

Table 4. Results when modelling the J.S. Bach Chorales dataset with two strong baselines and *TonicNet_Z* trained on various combinations and subsets of the JS Fake Chorales and J.S. Bach Chorales datasets. Rows marked with asterisks are taken directly from [8], while rows marked with a cross are taken from [20].

evaluation on the notes only, as is typical in experiments on the JSB Chorales, omitting elements containing chord tokens specifically included as part of the *TonicNet* representation when averaging the loss for each sample.

We can see that even without fine-tuning parameters, training the model only on the JS Fake Chorales obtains results close to training on the original set Bach chorales, whether augmented or not. Remarkably, we are even able to perform better than strong baselines [6, 20] without using any real training data. Naturally, there are far more samples in *JSF* than *Bach*, so we use *JSF-top* for a more balanced comparison and find that the model trained on this dataset indeed performs worse, but still within a close range compared to training on the canonical dataset. We see that there is no significant difference between using *JSF-top* and *JSF-top-s* during training, but that using either in combination with *Bach* provides better results than augmenting via *MM*. The dataset size is roughly doubled from the original in all three cases when using *JSF-top*, *JSF-top-s* and *MM*, meaning that the final training set size is close to equal for these cases, in turn suggesting that using the JS Fake Chorales is a higher quality form of augmentation than *MM*. We are therefore able to set a new state-of-the-art result of 0.208 validation set NLL on the JSB Chorales by combining the original training set with the JS Fake Chorales.

5. CONCLUSION

We develop an algorithm capable of generating chorales in the style of Bach and sample 500 pieces from it consecutively with no human intervention. We devise an experiment for human evaluation designed to be fair and free from bias, and through this find that these samples are perceived to be composed by Bach almost as readily as pieces which are in fact by Bach. In so doing, we collect data about human interaction with these pieces and create a dataset of 500 synthetic chorales with human annotation. By avoiding cherry-picking when conducting human evaluation, we demonstrate that the number of pieces in the JS Fake Chorales dataset could be readily scaled while expecting quality to be maintained. We further provide a web interface to readily enable such scaling of the dataset.

We run experiments to demonstrate the effectiveness of this dataset in a common MIR task. We show that training an algorithm on our dataset can be nearly as effective at modelling Bach's music than training on his actual pieces, outperforming several strong baselines. Finally, by combining both the JS Fake Chorales and Bach's chorales into a training corpus, we achieve state-of-the-art results on JSB Chorales dataset.

In general, the motivation for any field of research is to see it lead to real-world applications where it enables people to do what they previously could not, or to to do things with fewer resources than was previously possible. In order to see widespread adoption of generative music algorithms in consumer applications, they must consistently perform well in at least one relevant task. Such tasks in the symbolic music domain include, but are not limited to, melody generation, melody continuation, accompaniment generation, music in-painting and music style transfer.

Today, no products which enable the aforementioned tasks could truly be described as being widely-used, with the exception of arpeggiators, a family of rule-based tools which perform a specific and narrow kind of style transfer in the symbolic domain. Ultimately this is due to a lack of algorithms which perform with sufficient consistency and quality to provide meaningful value to potential users. One of the most promising ways to overcome this is to break the two bottlenecks mentioned in Section 1 and leverage learning algorithms.

Large datasets of music in varying styles, especially containing polyphonic music, will be needed to create learning algorithms that perform all of the above tasks reliably, but these are not easily or cheaply obtainable. Hence we proposed a synthetic dataset with human evaluation data and on-demand scalability as one way to alleviate this problem. Conducting a listening experiment on hundreds of consecutive output allowed us to obtain a good picture of where samples meet or fall short of typical consumer expectation. Newly-generated samples from the same algorithm can be expected to be similarly distributed in this respect, and the function of distance between these samples and consumer expectation, measured as described in Sections 3 and 4, can be modelled in a semi-supervised way in future works. The ability to model this gap gives us a quantitative way to measure it for new algorithms, ultimately allowing researchers to close it more efficiently.

6. ACKNOWLEDGEMENTS

This work was supported by the Polish National Centre for Research and Development under Grant POIR.01.01.01-00-0322/20 titled "Humtap - research and development of the technology for audio-video content generation with machine learning and social sharing".

7. REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2009, pp. 248–255.

[2] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE International Conference on Computer Vision*, December 2015.

[3] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *IJCV*, 2020.

[4] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: a steerable model for Bach chorales generation," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 1362–1371. [Online]. Available: http://proceedings.mlr.press/v70/hadjeres17a.html

[5] F. Liang, "Bachbot: Automatic composition in the style of bach chorale," Master's thesis, University of Cambridge, August 2016.

[6] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *Proceedings of the 18th International Society for Music Information Retrieval*, 2017.

[7] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proceedings of the29th International Conference on Machine Learning, ICML 2012*, vol. 2, 06 2012.

[8] O. Peracha, "Improving polyphonic music models with feature-rich encoding," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020, pp. 169–175.

[9] C. Payne. (2019, April) Musenet. [Online]. Available: openai.com/blog/musenet

[10] M. Mauch and S. Dixon, "Pyin: A fundamental frequency estimator using probabilistic threshold distributions." in *Proceedings of ICASSP*. IEEE, May 2014, pp. 659–663. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6853678

[11] J. Kim, J. Salamon, P. Li, and J. Bello, "Crepe: A convolutional representation for pitch estimation," in *Proceedings of ICASSP*, 04 2018, pp. 161–165.

[12] O. Peracha and S. Head, "Gankyoku: a generative adversarial network for shakuhachi music," in *Proceedings of the International Computer Music Conference*. ICMA, July 2019.

[13] Q. Kong, B. Li, J. Chen, and Y. Wang, "Giantmidi-piano: A large-scale MIDI dataset for classical piano music," *CoRR*, vol. abs/2010.07061, 2020. [Online]. Available: https://arxiv.org/abs/2010.07061

[14] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution piano transcription with pedals by regressing onsets and offsets times," *CoRR*, vol. abs/2010.01815, 2020. [Online]. Available: https://arxiv.org/abs/2010.01815

[15] C.-Z. A. Huang, C. Hawthorne, A. Roberts, M. Dinculescu, J. Wexler, L. Hong, and J. Howcroft, "The Bach Doodle: Approachable music composition with machine learning at scale," in *International Society for Music Information Retrieval (ISMIR)*, 2019. [Online]. Available: https://goo.gl/magenta/bach-doodle-paper

[16] J. Ens and P. Pasquier, "Mmm : Exploring conditional multi-track music generation with the transformer," 2020. [Online]. Available: https://arxiv.org/abs/2008.06048

[17] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T. Liu, "Musicbert: Symbolic music understanding with large-scale pre-training," 2021. [Online]. Available: https://arxiv.org/abs/2106.05630

[18] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data." in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, J. S. Downie and R. C. Veltkamp, Eds. International Society for Music Information Retrieval, 2010, pp. 637–642.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Proceedings of Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[20] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure." in *ICLR (Poster)*, 2019.

[21] K. Crowston, "Amazon mechanical turk: A research tool for organizations and information systems scholars," in *Shaping the Future of ICT Research. Methods and Approaches*, A. Bhattacherjee and B. Fitzgerald, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 210–221.

SOUND-ACCOMPANYING MOVEMENTS ENHANCE PERCEIVED AROUSAL IN MUSIC PERFORMANCE EVEN FROM A DISTANCE

Hanna Järveläinen
ICST Institute for Computer Music and Sound Technology
Zurich University of the Arts
hanna.jarvelainen@zhdk.ch

ABSTRACT

This paper describes an experiment measuring the impact of visual cues on the perception of music performance under varying static and dynamic visual conditions. Perceived arousal was measured in an audio-visual experiment using English horn performances. Audio and video recordings were made in a concert hall from short and long distance with and without sound-accompanying movements. Participants (N=32) rated perceived arousal continuously during watching, listening, or watching and listening. Sound-accompanying movements enhanced perceived arousal in audio-visual compared to auditory-only perception, while an immobile performance had a diminishing effect. In all sensory modalities, less arousal was perceived at longer distance. However, the positive effect of sound-accompanying movements in the visual channel was stronger than the negative effect of distance which is caused by both auditory and visual cues. Thus, through the visual channel, sound-accompanying movements may compensate for being seated far away from the performer at a concert.

1. INTRODUCTION

For each member of a concert audience, the experience is a unique combination of multisensory perception of music performance, transmission of the auditory, visual, and vibrotactile cues in the concert hall, and a specific social frame [1]. This study investigates the importance of visual information to the concert experience in an ecological setting but limited scope, focusing on audio-visual interactions in perceived arousal under varying visual conditions.

The high importance of visual cues to the perception of music performance is by now well known. In continuous measurements of musical tension, both auditory and visual channels were found to be important and reliable, although not necessarily correlated, sources of information [2]. The auditory channel is often thought to dominate; yet, mounting behavioral and physiological evidence shows that audio-visual perception of lower-level attributes such as tension and arousal is driven by both channels [3, 4].

Visual information also impacts many higher-level aspects of music performance. Visual kinematic cues may in fact communicate expressiveness and expressive intentions more effectively than auditory cues [5,6], possibly through coding loundess variation [7]. Visual information seems to weigh more in judgments of performance quality and expertise [8–11] and player identification [12]. Through body gestures, performers also communicate specific emotions [13] as well as their musical role in the ensemble [14].

In a live concert, both auditory and visual cues travel to the receiver over a distance. Research into the influence of visual conditions is limited in scope, however. Impact of visual cues has been shown on judgments of loudness and reverberance in auditoriums [15] and on room acoustic experience in virtual reality [16]. Static cues such as viewing angle, distance to stage, and size of photographic stage view were significant predictors of opera seat preference, although second to sound intensity [17]. Results from game research suggest that players experience greater arousal and immersion with a larger screen [18].

An experiment was carried out to explore the importance of visual cues under varying visual conditions in a concert hall performance. Earlier measurements show that participants perceive arousal in a coordinated way from both auditory and visual cues [4]. Two more independent variables were now added: observer's distance from the stage and the amount of sound-accompanying movements [19] exerted by the performer. This allowed separating the effects of static visual cues (viewing conditions) and dynamic visual cues (sound-accompanying movements). Response variable was perceived arousal, measured continuously over time [20, 21]. For better control over the experimental conditions, recorded performances were used.

2. EXPERIMENT

2.1 Stimuli, Design, and Participants

All stimuli consisted of a three-minute excerpt from the beginning of In Nomine for English horn by György Kurtág (1926-). The soloist, an early-career professional, was placed at center stage, ca 1 m from the front edge, in a 400-seat concert hall. Recordings were made with and without sound-accompanying movements from two positions in the concert hall as presented in Figure 1. Audio-

Copyright: © 2022 Hanna Järveläinen et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

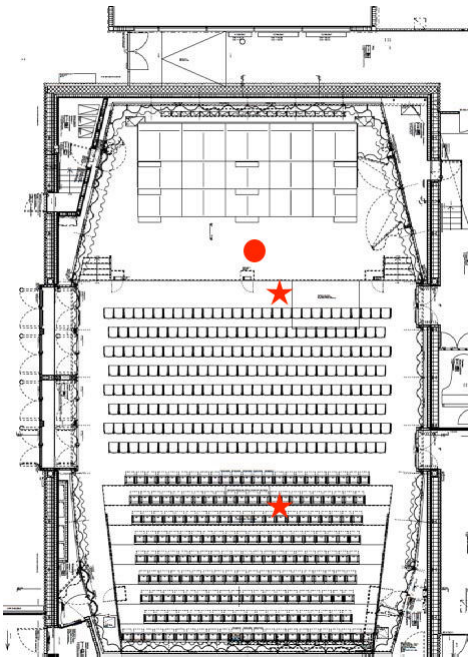


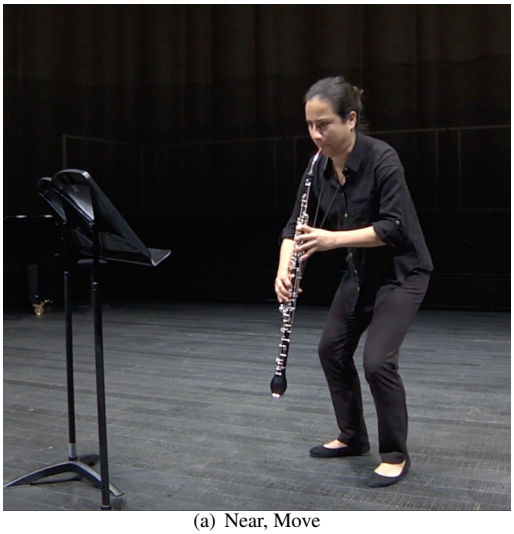
Figure 1. Performer (•) and recording (⋆) positions in the concert hall.

video recordings were made using one video camera and a stereo microphone pair placed at camera position. The relative differences in sound intensity and size of the stage view caused by the near and far recording positions were preserved.

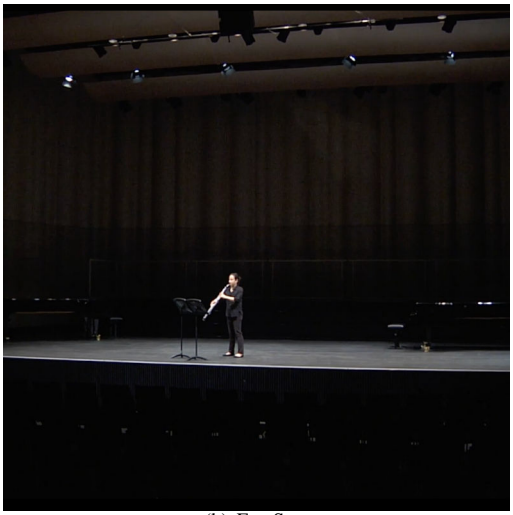
Three predictors (independent variables) were manipulated in the experimental design as follows:

- Distance with two levels: [Near, Far]. The Near recordings were made from the first row, at a distance of ca 2.7 m to the performer. The Far recordings were made from row 11 at a distance of ca 13 m.
- Movement with two levels: [Move, Stay]. In the Move condition, the performer exerted sound-accompanying movements in addition to necessary sound-producing gestures. She was instructed to use her own movement repertoire in a way that would be congruent with the tempo and loudness variation of the music. The Stay condition consisted of sound-producing gestures without intentional sound-accompanying movements.
- Modality with three levels: [Auditory only (A), visual only (V), audio-visual (AV)]. The A condition consisted in audio recordings without video, the V condition in silent video recordings, and the AV condition in synchronized and congruent audio and video recordings.

N=32 subjects took part in the experiment (15 F / 17 M, aged M=24, sd=2.6; all but one were music students). Distance was a nested factor so that half of the participants rated Near recordings and the other half rated Far recordings. Within the Distance groups, Movement and Modality were crossed such that each participant rated all six resulting combinations: A-Move, A-Stay, AV-Move, AV-Stay, V-Move, and V-Stay. This grouping was a compromise between recruitment efforts, effective data collection, and reduced repetition of the material. The Near and Far groups were separated in order to prevent the possibility of forming expectations based on the other distance condition.



(a) Near, Move



(b) Far, Stay

Figure 2. Still images from the videos in two of the four conditions. Images are cropped and downsized but relative image sizes are preserved.

2.2 Measurement

Participants rated perceived arousal during watching and/or listening. Perceived arousal was defined as a measure of how much energy, activity, or tension the performance contains at any given moment. Audio was presented through headphones and video on a 24-inch monitor in full-screen mode. Participants sat at a normal working dis-

tance from the screen. The six three-minute stimuli were presented in randomized order, and participants gave responses by adjusting a slider continuously throughout the excerpts. The range of the slider end points was [0,1], and the values were read into a time series at a frequency of 4 Hz.

2.3 Data Pre-Processing

The raw measurement data were registered in order to compensate for slight differences in tempi and length of section breaks in the stimuli. The performer played systematically slightly faster and took shorter breaks between sections when adding sound-accompanying movements. Within sections the tempi were stable, however. The recordings were annotated manually according to ten sections and two major section breaks, and the time scales of all measurements were linearly adjusted so that section durations matched to the slowest stimulus which was the Near-Stay version. The length of each registered time series was again cut to 180 seconds.

2.4 Statistical Model

The data were analysed by means of functional analysis of variance (fANOVA), a technique suited to describe variability in a functional (time-varying) response curve in terms of predictors that can be either curves or scalars [22]. A functional additive regression model was constructed that contains both functional and scalar components for each predictor, decomposing the effects of the predictors into curves and constant shifts as follows:

$$\begin{aligned} \text{y(t)} \sim & \text{f.general(t)} + \\ & \text{f.modality(t)} + \text{f.movement(t)} + \text{f.distance(t)} + \\ & \text{modality} + \text{movement} + \text{distance} + \\ & \text{modality:movement} + \text{modality:distance} + \\ & \text{movement:distance} \end{aligned} \tag{1}$$

where

- y(t) is functional perceived arousal
- f.general(t) is a common functional component
- f.modality(t), f.movement(t), and f.distance(t) are effects of functional predictors
- modality, movement, and distance are effects of scalar predictors
- : marks interactions between scalar predictors

The model was fit by approximate Bayesian inference in R [23] using integrated nested Laplace approximation available in the INLA package [24]. INLA is a fast and computationally effective alternative to Markov chain Monte Carlo sampling and can be used to solve temporal additive regression models [25].

3. RESULTS

Figures 3 and 4 present the estimated effects of the predictors in Eq. 1. The top panel of Figure 3 shows the general time-varying component, the mid panel the effects of modality, and the bottom panel the combined general trend and modality effects. It is seen that audio-visual presentation has a general slight positive effect compared to both unimodal auditory or visual conditions.

Figure 4 shows the remaining effects of movement and distance after the general trend and modality effects have been extracted. Differences are mainly caused by distance in the auditory only condition and by movement in the visual only condition. In audio-visual perception, both effects are equally present. The combined effect is the most negative for the Stay-Far combination, while the Stay-Near and Move-Far combinations result in similar net effects, second only to the Move-Near combination.

The complete model fit combines the effects shown in Figures 3 and 4. To investigate the significance of differences caused by various effect combinations, the remaining analysis will focus on the fitted posterior mean arousal functions and their 95% confidence bands, expressed as the range between the 2.5% and 97.5% quantiles of the estimated means. Differences between conditions are significant at instances where the respective confidence bands do not overlap.

Figure 5 presents the 95% confidence bands for the fitted means (grey bands) and the measured mean ratings (thin lines). The measured means are within the fitted confidence bands with small and momentary deviations. An exception is the first minute of the Near-Move-V condition, where the additive functional components do not entirely capture the perceived changes.

The findings seen in Figure 4 are shown in full context in Figures 5 and 6. They present the same data faceted differently: the former highlights the effect of movement and the latter the effect of distance. In the auditory-only condition, the confidence bands for Move and Stay conditions overlap entirely for both distance conditions in Figure 5, indicating that sound-accompanying movements do not influence auditory ratings. Auditory perceived arousal is slightly diminished by long distance as seen in Figure 6.

In the visual-only condition, distance has a similar, rather weak negative effect (Figure 6). In contrast, the difference between Move and Stay conditions in Figure 5 is large and the respective confidence bands do not overlap at all. Thus, the visual channel effectively conveys arousal information from sound-accompanying movements.

Influence of the visual channel is evident in the audio-visual condition, where the confidence bands for Move and Stay conditions in Figure 5 only overlap in the time period 120-160 s and 150-160 s for the Near anf Far conditions, respectively. For the most part, sound-accompanying movements significantly enhance audio-visual perceived arousal through the visual channel, independently of observer's distance to the performer.

Finally, Figure 7 presents the interesting comparison between auditory-only and audio-visual perception. In presence of sound-accompanying movements, more arousal is

Figure 3. General trend and effects of modality from the statistical model (Eq. 1).

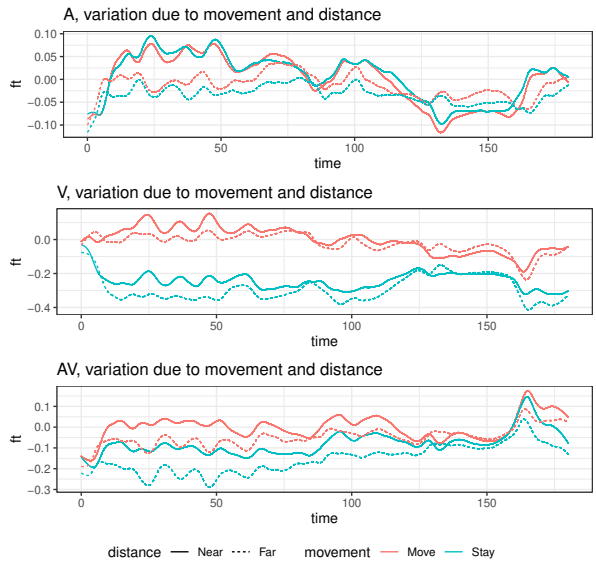


Figure 4. Effects of movement and distance from the statistical model in auditory only, visual only, and audio-visual conditions (Eq. 1).

perceived in the audio-visual condition, while in their absence, the opposite is true. Distance influences these results, however. In the Near-Move condition where visual cues are the most salient, audio-visual perception is significantly increased from the audiotory-only condition almost throughout the stimulus. In the Far-Move condition, however, the difference is significant only momentarily. Similarly, in the Far-Stay condition where visual cues are the least salient, audio-visual perception is significantly diminished from the auditory-only condition and less so in the Near-Stay condition.

The above analysis focused on significant level differences between conditions. Another important aspect is the amount of perceived changes, described by variability over time. It is an indicator of how much information the stimulus conveys over each sensory modality.

For a rough comparison of variability, the boxplots in Figure 8 present the interquartile ranges of the fitted functional means for each stimulus. (This analysis excludes the first 10 seconds during which the rating profiles rise from zero.) In the auditory-only condition, variability is unaffected by added movements and slightly diminished by long distance. In the visual-only condition, the effect of distance is similar. However, variability is notably higher in presence of sound-accompanying movements, exceeding even the auditory-only condition. The joint effect in the audio-visual condition is a slightly higher variability with movements compared to auditory-only perception; lack of movements in turn does not notably affect variability.

The audio-visual enhancement can thus be observed in both mean arousal levels and perceived changes over time [1] .

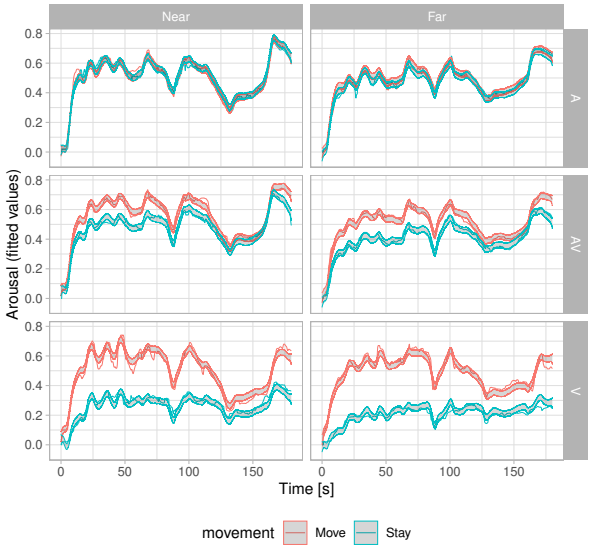


Figure 5. Effect of movement on perceived arousal. Grey bands: 95% functional confidence bands for fitted means; thin lines: measured means.

[1] Dataset and analysis are available at: https://doi.org/10.5281/zenodo.6461144

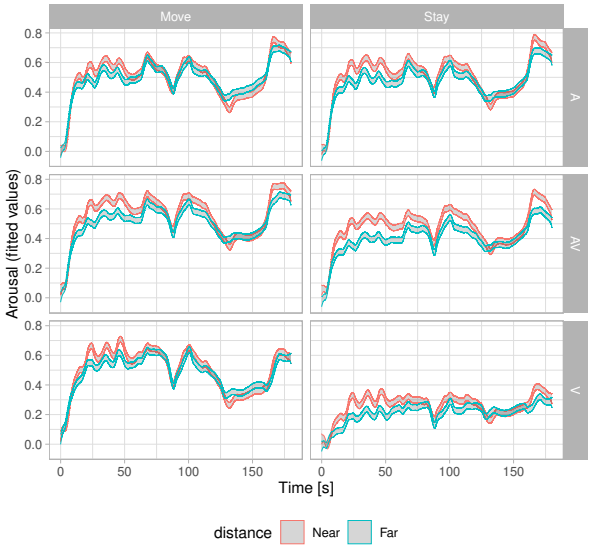


Figure 6. Effect of distance on perceived arousal. Grey-filled bands: 95% functional confidence bands for fitted means.

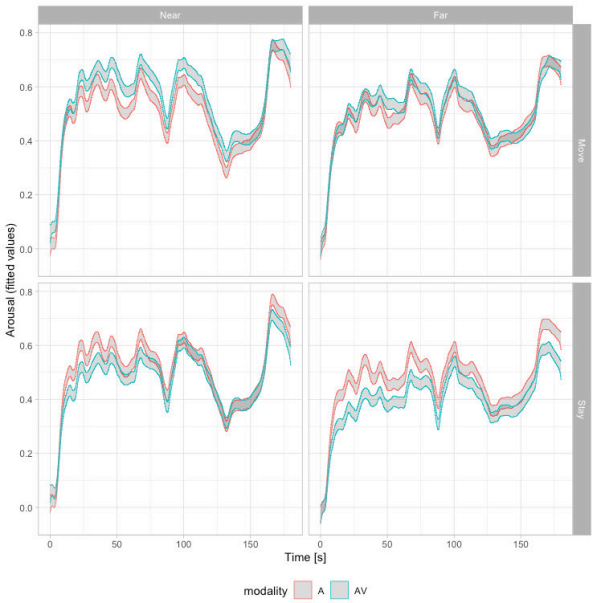


Figure 7. Comparison between audio-only and audio-visual conditions: 95% functional confidence bands for fitted means.

4. DISCUSSION

In this experiment, dynamic visual cues were varied through the amount of sound-accompanying movements. These movements did not affect perceived arousal in the auditory-only condition. Through visual cues, however, they significantly enhanced perceived arousal in audio-visual perception; respectively, their absence had a diminishing effect. The significance of visual cues is in line with earlier results on audio-visual interactions in perception of changes in expressive intentions [6].

Static visual cues were varied through distance to the performer. The main effect of distance was similar in all sensory modalities (A / AV / V): slightly higher average arousal and more changes were perceived near the performer, both in absence and presence of movements. This was expected, as both auditory and visual cues are most salient at short distance. For auditory cues, this may be explained through concert hall acoustics. In the reverberant field, sound intensity decays in a concert hall somewhat, but not as much as it would with distance in free field. The amount of motion in the video recordings is in turn directly connected to distance through image size, and kinematic cues as well as facial expressions are more visible from close distance. Importantly however, perceived arousal in the visual-only condition did not decrease dramatically with distance; the results might be different if physiological responses were measured.

The joint effect of static and dynamic visual cues was the most negative at long distance without movements, where perceived arousal was significantly diminished in audio-visual compared to auditory-only perception (Figure 7, lower right panel). This loss was compensated for by added movements (Figure 7, upper right panel). It is of course a limitation that both distance and movement were varied on only two levels; in a live concert, these both vary at a wide range with seating and performance style. However, rather than short distance to the stage, presence of salient dynamic visual cues seems to be key. Immobile performance, instead of being ignored, leads to reduced audio-visual perception. In such cases, concert-goers may easily optimize their perceived arousal by closing their eyes – at the cost of losing visual information entirely, of course.

In audio-visual perception, congruent sound-accompanying movements enhanced perceived variation in arousal even though no effect was present in auditory-only perception. As perceived changes in arousal correlate with loudness/intensity changes over time [26], a similar enhancement could be achieved by presenting auditory-only stimuli with enhanced dynamic range. On the other hand, studies on perceived quality of concert hall acoustics show that preferred halls reproduce loudness variation especially well [27]. An interesting continuation for this research might be to explore cross-modal enhancement of perceived concert hall acoustics by adding salient and congruent visual stimuli.

4.1 Limitations

Participants in this study were musically trained with one exception. Adding a group of non-musicians as participants would produce important knowledge about differences between these groups. In auditory perception, previous research reports very similar emotional responses of both musicians and non-musicians to measures including musical tension and magnitude of esthetic experience [28–30]. However, musical training has been shown to alter both the behavioural and neural processing of sadness and fear [31] and the emotional impact of tempo differences [32]. In visual or audio-visual perception, no effects of musical training were observed on perceived ten-

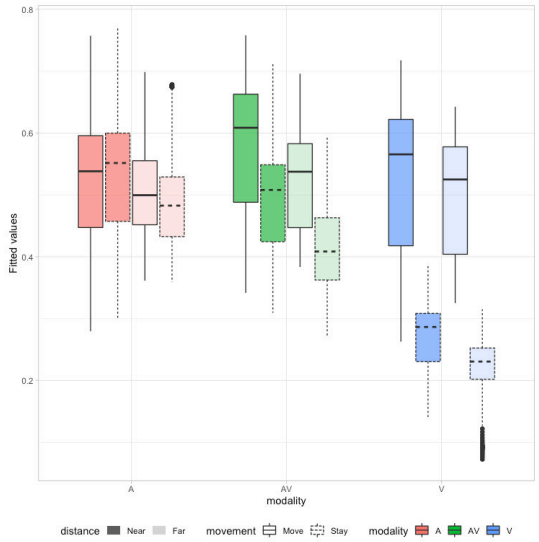


Figure 8. Boxplots of fitted values for all factor combinations. Interquartile ranges of the fitted functional means allow a rough comparison of perceived changes for each stimulus.

sion [33]. In a study on perceived effort and arousal, musicians gave slightly higher ratings in the auditory only condition and non-musicians in the visual only condition, although the analysis was not conclusive [4]. Claims have been made, however, that musically unskilled listeners perceive the performer's emotional intentions entirely through visual cues [34]. This evidence leads to a future hypothesis that non-musicians might rate the auditory channel rather similarly but perhaps rely more on visual information, which would rather emphasise the effect reported in the current study.

For the sake of systematic control of the conditions, the current results were obtained from recordings observed in a laboratory setting. Extrapolating them to witnessing a live performance is of course difficult. A live performance is thought to enhance the experience in many ways [35], and previous research has shown higher emotional convergence between audience members in a live concert [36]. Less is known about the perception of basic emotional dimensions such as arousal in live versus recorded performances. The basic trends should hold in both cases: perceived arousal decreases with decreasing loudness and slightly with decreasing image size, and congruent movements may enhance it through the visual channel. At each seat position in the concert hall, these factors have an individual impact. The resulting net effect should therefore also vary almost continuously.

Objective acoustic and visual analysis of the stimuli was not part of this investigation. In an earlier dataset, collected from sound-producing movements in cello performance [4], ratings correlated highly with simple predictors: auditory ratings with RMS energy, roughness, and event density, and visual ratings with the amount of motion (expressed as root-mean-square of differenced head and finger positions). A deeper analysis should include these time-varying predictors as well as measures of facial expressions and muscle tension. This work, along with comparison of models including interactions between functional predictors, is left for a future investigation.

5. CONCLUSIONS

It was shown how sound-accompanying movements can enhance, and how their absence can diminish audio-visual perceived arousal in music. In both sensory modalities, higher average arousal and more changes were perceived near the performer than further away. In audio-visual perception the effect of distance, caused by changes in both auditory and visual cues, may be compensated for by the positive effect of sound-accompanying movements. These results add to the mounting body of evidence that visual information influences our perception of music performance in numerous ways.

Acknowledgments

Angela Calvo Rios and Mario Bruderhofer are gratefully acknowledged for performing and recording the stimuli.

6. REFERENCES

[1] M. Wald-Fuhrmann, H. Egermann, A. Czepiel, K. O'Neill, C. Weining, D. Meier, W. Tschacher, F. Uhde, J. Toelle, and M. Tröndle, "Music Listening in Classical Concerts: Theory, Literature Review, and Research Program," *Front. Psychol.*, vol. 12, p. 1324, apr 2021.

[2] B. W. Vines, C. L. Krumhansl, M. M. Wanderley, and D. J. Levitin, "Cross-modal interactions in the perception of musical performance." *Cognition*, vol. 101, no. 1, pp. 80–113, 2006.

[3] C. Chapados and D. J. Levitin, "Cross-modal interactions in the experience of musical performances: Physiological correlates," *Cognition*, vol. 108, no. 3, pp. 639–651, sep 2008.

[4] H. Järveläinen, "Audiovisual perception of arousal, valence, and effort in contemporary cello performance," in *Proceedings of the Sound and Music Computing Conference*, Malaga, Spain, 2019, pp. 511 – 518.

[5] J. Vuoskoski, M. Thompson, E. Clarke, and C. Spence, "Crossmodal interactions in the perception of expressivity in musical performance." *Atten. Percept. Psychophys.*, vol. 76, no. 2, pp. 591–604, 2014.

[6] B. W. Vines, C. L. Krumhansl, M. M. Wanderley, I. M. Dalca, and D. J. Levitin, "Music to my eyes: Cross-modal interactions in the perception of emotions in musical performance," *Cognition*, vol. 118, no. 2, pp. 157–170, feb 2011.

[7] J. Vuoskoski, M. Thompson, C. Spence, and E. Clarke, "Interaction of sight and sound in the perception and experience of musical performance," *Music Percept.*, vol. 33, no. 4, 2016.

[8] N. K. Griffiths and J. L. Reay, "The relative importance of aural and visual information in the evaluation of western canon music performance by musicians and nonmusicians," 2018.

[9] C.-J. Tsay, "Sight over sound in the judgment of music performance." *Proc. Natl. Acad. Sci. USA*, vol. 110, no. 36, pp. 14 580–14 585, 2013.

[10] M. Rodger, C. Craig, and S. O'Modhrain, "Expertise is perceived from both sound and body movement in musical performance," *Hum. Mov. Sci.*, vol. 31, no. 5, 2012.

[11] F. Platz and R. Kopiez, "When the eye listens: a meta-analysis of how audio-visual presentation enhances the appreciation of music performance," *Music Percept.*, vol. 30, no. 1, pp. 71–83, 2012.

[12] H. F. Mitchell and R. a. R. MacDonald, "Listeners as spectators? Audio-visual integration improves music performer identification," *Psychol. Music*, vol. 42, pp. 112–127, 2012.

[13] S. Dahl and A. Friberg, "Visual perception of expressiveness in musicians' body movements," *Music Perc.*, vol. 24, pp. 433–454, 2007.

[14] M. B. Kuessner, K. Kuessner, E. V. Dyck, and B. Burger, "All eyes on me: Behaving as soloist in duo performances leads to increased body movements and attracts observers' visual attention," 2020.

[15] Y. Tokunaga, D. Okuie, and T. Terashima, "Influence of visual information on sound evaluation in auditorium," in *Proc. Meet. Acoust.*, vol. 19, 2013, p. 40104.

[16] B. N. J. Postma and B. F. G. Katz, "The influence of visual distance on the room-acoustic experience of auralizations," *Cit. J. Acoust. Soc. Am.*, vol. 142, p. 15008, 2017.

[17] J. Y. Jeon, Y. H. Kim, D. Cabrera, and J. Bassett, "The effect of visual and auditory cues on seat preference in an opera theater," *J. Acoust. Soc. Am.*, vol. 123, no. 6, pp. 4272–4282, jun 2008.

[18] J. Hou, Y. Nam, W. Peng, and K. M. Lee, "Effects of screen size, viewing angle, and players' immersion tendencies on game experience," *Comput. Human Behav.*, vol. 28, pp. 617–623, 2012.

[19] R. I. Godøy and M. Leman, Eds., *Musical gestures : sound, movement, and meaning*. Routledge, 2010.

[20] J. A. Russell, "A circumplex model of affect." *J. Pers. Soc. Psychol.*, vol. 39, no. 6, pp. 1161–1178, 1980.

[21] M. Zentner and T. Eerola, "Self-report measures and models," in *Handbook of Music and Emotion: Theory, Research, Applications*, P. Juslin and J. Sloboda, Eds. Oxford University Press, 2010.

[22] J. O. Ramsay and B. W. Silverman, *Functional Data Analysis*, ser. Springer Series in Statistics. New York, NY: Springer New York, 2005.

[23] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2021. [Online]. Available: https://www.R-project.org/

[24] H. Rue, S. Martino, and N. Chopin, "Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations," *J. R. Stat. Soc. Ser. B (Statistical Methodol.*, vol. 71, no. 2, pp. 319–392, apr 2009.

[25] "Functional ANOVA using INLA," https://www.r-bloggers.com/2012/02/ functional-anova-using-inla-update/, note = .

[26] R. T. Dean, F. Bailes, and E. Schubert, "Acoustic intensity causes perceived changes in arousal levels in music: an experimental investigation." *PLoS One*, vol. 6, no. 4, p. e18591, apr 2011.

[27] J. Pätynen and T. Lokki, "Perception of music dynamics in concert hall acoustics," *J. Acoust. Soc. Am.*, vol. 140, no. 5, pp. 3787–3798, 2016.

[28] C. K. Madsen, "Emotional response to music," *Psychomusicology*, pp. 59–67, 1997.

[29] W. E. Fredrickson, "Perception of tension in music: musicians versus non musicians," *J. Music Ther.*, vol. 37, no. 1, pp. 40–50, 2000.

[30] E. Bigand, S. Vieillard, F. Madurell, J. Marozeau, and A. Dacquet, "Multidimensional scaling of emotional responses to music: The effect of musical expertise and of the duration of the excerpts," *Congition and Emotion*, vol. 19, no. 8, pp. 1113–1139, 2005.

[31] M. Park, E. Gutyrchik, Y. Bao, Y. Zaytseva, P. Carl, L. Welker, E. Pöppel, M. Reiser, J. Blautzik, and T. Meindl, "Differences between musicians and non-musicians in neuro-affective processing of sadness and fear expressed in music." *Neuroscience Letters*, vol. 566, pp. 120–4, 2014.

[32] Y. Liu, G. Liu, D. Wei, Q. Li, G. Yuan, S. Wu, G. Wang, and X. Zhao, "Effects of Musical Tempo on Musicians' and Non-musicians' Emotional Experience When Listening to Music," *Frontiers in Psychology*, vol. 9, p. 2118, 2018.

[33] R. J. D. Frego, "Effects of Aural and Visual Conditions on Response to Perceived Artistic Tension in Music," *J. Res. Music Educ.*, vol. 47, no. 1, pp. 31–43, 1999.

[34] J. Davidson and J. S. Correia, "Body movement," in *The Science Psychology of Music Performance – Creative Strategies for Teaching and Learning*, R. Parncutt, Ed. Oxford: Oxford University Press, 2002.

[35] L. Finnäs, "Presenting music live, audio-visually or aurally: does it affect listeners' experiences differently?" *Br. J. Music Educ.*, vol. 18, no. 1, pp. 55–78, 2001.

[36] E. Coutinho and K. R. Scherer, "The effect of context and audio-visual modality on emotions elicited by a musical performance," *Psychol. Music*, vol. 45, no. 4, pp. 550–569, 2017.

Replicating Human Sound Localization with a Multi-Layer Perceptron

Eric Michael Sumner
University of Iceland
ems36@hi.is

Runar Unnthorsson
University of Iceland
runson@hi.is

Morris Riedel
University of Iceland & Jülich Supercomputing Centre
morris@hi.is

ABSTRACT

One of the key capabilities of the human sense of hearing is to determine the direction from which a sound is emanating, a task known as localization. This paper describes the derivation of a machine learning model which performs the same localization task: Given an audio waveform which arrives at the listener's eardrum, determine the direction of the audio source. Head-related transfer functions (HRTFs) from the ITA-HRTF database of 48 individuals are used to train and validate this model. A series of waveforms is generated from each HRTF, representing the sound pressure level at the listener's eardrums for various source directions. A feature vector is calculated for each waveform from acoustical properties motivated by prior literature on sound localization; these feature vectors are used to train multi-layer perceptrons (MLPs), a form of artificial neural network, to replicate the behavior of single individuals. Data from three individuals are used to optimize hyperparameters of both the feature extraction and MLP stages for model accuracy. These hyperparameters are then validated by training and analyzing models for all 48 individuals in the database. The errors produced by each model fall in a log-normal distribution. The median model is capable of identifying, with 95% confidence, the sound source direction to within 20 degrees. This result is comparable to previously-reported human capabilities and thus shows that an MLP can successfully replicate the human sense of sound localization.

1. INTRODUCTION

The Acoustic and Tactile Engineering Lab (ACUTE) of the Icelandic EuroCC National Competence Center [1] for Artificial Intelligence and High-Performance Computing performs research and product development for societally relevant challenges in many applications together with its European partners (e.g., project *Sound of Vision* won the Tech for Society award in 2018 [2]). This includes the development of wearable assistive devices for visually impaired persons, cochlear implant recipients, and solutions for delivering accurate virtual acoustics.

One particular application of virtual acoustics is the development of spatial audio systems, which are designed to present realistic virtual soundscapes to the listener. In order for a spatial audio system to present a realistic soundscape, it is imperative to stimulate all aspects of an individual's sense of hearing. One important aspect of this is *'localization'*, the ability of a listener to determine the location of a sound source. This requires an accurate characterization of the listener's head-related transfer function (HRTF). A number of approaches have been proposed to estimate these individualized HRTFs, such as selecting a nearest match from a database of measured HRTFs [1] or calculating the HRTF from a numerical simulation of the indivdual's head shape [2]. Unfortunately, they all produce results that are significantly worse than a direct measurement taken in an anechoic chamber [3]. Developing a better HRTF estimation method for use in spatial audio systems is an ongoing project within ACUTE.

A major challenge in this project is evaluating the effectiveness of the HRTF estimates produced. One approach is to develop a model which can replicate the localization response of an individual to arbitrary audio waveforms. This model can then be presented with the waveforms produced by a candidate spatial audio system, and the intended virtual location compared to the model's output, i.e. the listener's perceived audio source direction.

This paper derives a class of individualized machine learning models for this task, based on a multi-layer perceptron (MLP) [4]. The training data for each model consists of an audio signal dataset derived from a white noise generator and an HRTF from the ITA-HRTF database [5], representing sound pressure levels (SPLs) at the individual's eardrums. Features representing both broadband and spectral interaural level difference (ILD) and interaural time delay (ITD) information are extracted from these waveforms and presented to the input layer of an MLP.

Three of these individualized models are simultaneously optimized for both accuracy and training cost, producing a generally-applicable hyperparameter configuration that can be used to train models against arbitrary HRTFs. Models are then trained for the remaining 45 individuals in the ITA-HRTF dataset to validate the hyperparameter configuration's general applicability.

The remainder of this paper is structured as follows: Section 2 provides a brief introduction to the computational models employed and the physical mechanism of sound localization. Section 3 describes the dataset, hardware, and software used for this experiment. Section 4 derives the structure of an individualized sound localization model and

[1] http://ihpc.is/community
[2] https://soundofvision.net/sound-of-vision-at-ict-2018/

Copyright: © 2022 Eric Michael Sumner et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

describes the hyperparameter optimization process. Individualized models are then trained for 45 additional HRTFs that were not used in the derivation process; Sec. 5 presents an analysis of these models, including overall statistics and a detailed investigation of selected models. The paper concludes with some remarks about the the possible applications of these models in Sec. 6.

2. BACKGROUND

This section provides an introduction to the underlying concepts of sound localization and machine learning. Section 2.1 describes the physical mechanism of sound localization. Section 2.2 then provides a brief overview of the machine learning techniques used, i.e. multi-layer perceptrons and hyperparameter optimization.

2.1 Sound Localization

Sound localization is possible due to the acoustic filtering effected by sound waves interacting with an individual's body, especially the pinnae [6]. As incoming sound waves interact with the pinna, reflections and refractions internal to the pinna mutually interfere, altering the waveform that arrives at the listener's eardrum as illustrated in Fig. 1a. For point sources in the far field, this alteration can be represented as a direction-dependent, linear and time-invariant audio filter known as the HRTF.

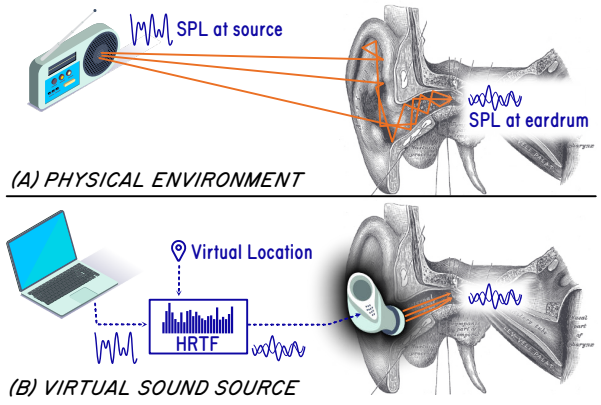


Figure 1. Acoustic filtering of the pinna and its relationship to the HRTF. Adapted from Gray, Plate 907 [7]

As the geometry of each individual's pinnæ are unique, the HRTF is also unique to each individual; determining an analytic relationship between these is a field of active research. With an accurate measurement of an individual's HRTF, it is possible to predict the SPL at each eardrum given a source audio signal and its location, as illustrated in Fig. 1b. When this prediction is played through headphones, bypassing the physical filter, the listener perceives the sound to be coming from the direction of the virtual source [8].

Prior studies have identified the acoustic properties necessary for humans to perform sound localization tasks. For sound sources on the horizontal plane, humans rely primarily on the ITD and ILD [9]. Vertical localization additionally requires spectral information in the 4-16 kHz range [10]. The frequency response of a typical HRTF exhibits a number of notches in this range, with directionally-varying center frequencies. These notches introduce local spectral gradients which the human auditory system uses to estimate the source direction [11].

2.2 Machine Learning

'Machine learning' refers to the class of algorithms that can infer structure from examples, instead of having that structure explicitly stated by a programmer. One of the earliest of these algorithms is still in common use today, the multi-layer perceptron; i.e. a fully-connected, feed-forward artificial neural network, which is often coupled to a domain-specific feature extraction stage.

An MLP is formed of an input layer, several hidden layers, and an output layer, each consisting of several *'neurons'*, or *'nodes'*. The output of each neuron is defined as a linear combination of the previous layer's output values passed through a nonlinear, univariate *'activation function'*. The linear input weights of all the neurons in the network form the parameters that will be learned during training. Training an MLP is a supervised process which requires exemplar data samples labeled with a desired output. Each example is presented to the network's input layer and the corresponding output layer values are compared with the example's desired output. A backpropagation process then determines the contribution of each parameter to this measured error. The error contribution for several examples are grouped together in a *'batch'*, and a gradient descent optimization algorithm uses this information to adjust the internal parameters such that the observed errors approach zero. In this way, the entire *'training set'* of examples is processed many times in so-called epochs, ultimately resulting in an MLP which can reproduce the sample outputs with high accuracy.

Though MLPs are universal estimators on their own, they often require large amounts of computation and training data to produce an acceptable model. When some domain-specific information is known, adding a feature extraction stage prior to the MLP can greatly reduce these costs. Instead of feeding the source data directly into the input layer of the MLP, interesting properties of the source data are calculated and formed into a feature vector. This feature vector usually has a much lower dimensionality than the raw source data, which means that fewer neurons are required within the MLP. This, in turn, means that fewer internal parameters need to be calculated during training.

Most machine learning models have a number of parameters that cannot be automatically learned from the training data. These are called *'hyperparameters'* to indicate that they need to be specified manually before training begins. These can include properties of both the domain-specific feature extraction algorithm and the MLP itself, such as the included features, the topology of the MLP's network, the activation function of each neuron, the concrete optimization algorithm used, and the halting condition, among others.

Choosing appropriate values for these hyperparameters is the primary task of a machine learning engineer, but they often have subtle and non-intuitive effects on the overall

model accuracy. In many cases, the only way to select these is a process of trial-and-error. As the number of hyperparameters increases, performing this search manually becomes intractable. Recently, a number of automated approaches to this *'hyperparameter optimization'* problem have been proposed. These use techniques like Bayesian optimization to select the optimal hyperparameter configuration to test in each trial [12].

3. EXPERIMENTAL SETUP

The model training and evaluation code are written in Python and executed in the Jupyter programming environment [13] on a 6-core Intel i7-10750H CPU. Section 3.1 describes the dataset used for this experiment, and Sec. 3.2 describes the software environment used.

3.1 Dataset

The HRTFs used for this paper come from the ITA-HRTF database [5]. This database contains head geometry information and HRTFs for 48 individuals, stored in Spatially-Oriented Format for Acoustics (SOFA) [14] files. These files consist of impulse responses sampled along azimuth, elevation, and time dimensions. Each HRTF contains $360°$ of azimuth data and $160°$ of elevation data, sampled in $5°$ increments. These 2,304 impulse responses each contain 256 samples at 44.1 kHz, for an overall frequency resolution of 172.3 Hz.

3.2 Software Environment

All of the third-party packages in this paper use NumPy [15] arrays as their data transfer format. NumPy also provides the linear algebra, random number generation, and discrete Fourier transform (DFT) [16] routines needed for the model. Additional signal processing routines are provided by Librosa [17], which is designed for audio feature extraction.

The SOFA standard specifies that files shall be in network common data form [3] (netCDF) and mandates the inclusion of metadata describing the data collection process, such as the sampling rates and properties of the emmitters and recievers used. The implementation uses the PySofaConventions [4] package to make SOFA data available as NumPy arrays.

The MLP training and evaluation routines are provided by Scikit-learn [18], and the hyperparameter optimization routines are provided by the Optuna framework [19]. The plots in this report were generated with the Plotly package for Python [5].

4. MODEL DESIGN

This section describes a class of machine learning models, each of which can replicate a single individual's sound localization behavior; Fig. 2 shows a schematic representation of the model's operation and how it relates to the HRTF data used for training and evaluation. Section 4.1 describes the overall structure and training strategy, and Sec. 4.2 details the calculation of the feature vector. Section 4.3 describes the search used to determine appropriate values for the model's hyperparameters, shown in Fig. 2 in italic.

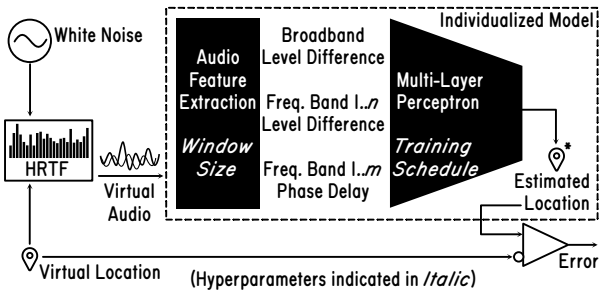


Figure 2. Structure of an individualized model and its relationship to training data

4.1 Model Design and Evaluation

Each model is trained on audio samples transformed by a single HRTF, to replicate the behavior of that individual. All of the source directions $\hat{y}$ present in the HRTF are extracted, and 20% are reserved for the testing (evaluation) set $\mathbf{E}$; the remaining 80% form the training set $\mathbf{T}$.

For each impulse response in the HRTF, NumPy generates half a second of white noise, which is then convolved with the impulse response to produce the virtual waveform which would be present at each of the listener's eardrums. These waveforms are then divided into windows and a feature vector $\vec{x}_i$ is calculated for each window i, as detailed in Sec. 4.2. This feature vector is then presented to the input layer of an MLP with three output nodes representing a vector $\vec{y}_i^*$ in Cartesian coordinates. The error $\Delta\vec{y}_i^*$, which is driven towards zero by the gradient descent optimization algorithm, is calculated as the difference between this estimate and the unit vector $\hat{y}_i$ which represents the true direction to the sound source (Eq. 1).

$$\Delta\vec{y}_i^* = \vec{y}_i^* - \hat{y}_i \tag{1}$$

The number of input nodes for the MLP is dependent on the hyperparameters of the feature extraction stage, which determine the length of the feature vector $\vec{x}_i$. The MLP has a total of five hidden layers; the first contains 128 nodes, which was chosen to maintain an approximate factor of two reduction from the input layer. Each subsequent hidden layer is a factor of two smaller, and the final hidden layer contains eight nodes which feed the three output nodes $\vec{y}^*$.

The input and output nodes use a linear activation function, and all of the hidden nodes use a common activation function f_{hidden}, which is a hyperparameter of the model; this is one of the hyperbolic tangent, the logistic sigmoid function, or the rectified linear unit (ReLU) (Eq. 2).

$$f_{\text{hidden}}(x) \in \left\{ \tanh(x),\ \frac{1}{1+e^{-x}},\ \max(0,x) \right\} \tag{2}$$

[3] https://www.unidata.ucar.edu/software/netcdf/

[4] https://andresperezlopez.github.io/pysofaconventions/

[5] https://plotly.com/python/

The MLP is trained with the adaptive moment estimation (Adam) [20] gradient descent optimization algorithm with L_2 paramater regularization, which introduces three additional hyperparameters: The L_2 penalty coefficient α and the Adam moment decay rates β_1 and β_2. This processes the entire training set $\mathbf{T}$ each epoch, in mini-batches of 200 data points. The training process terminates once the overall training loss fails to improve in 20 consecutive epochs.

As each window of audio is processed separately, the sequence of model outputs over time can be considered a signal of instantaneous direction estimates. In the case where neither the sound source nor the listener are moving, as here, the mean of this signal can be used as an overall direction estimate. It is also desirable for the model to be consistent, i.e. most directions should produce good estimates. Taking these concerns into consideration, the accuracy score A for the model is defined as the 95th percentile of the mean subtended error angle for all source directions in the testing set $\mathbf{E}$ (Eq. 3). Here, $\vec{y}_i^*$ is the estimated direction for audio window i, n is the set cardinality function, P_{95} is the 95th percentile function, and $\cdot$ is the scalar product.

$$A = P_{95}\left(\left\{\frac{\sum_{i\in\{j:\hat{y}_j=\hat{y}\}}\arccos\left(\frac{\vec{y}_i^*\cdot\hat{y}}{||\vec{y}_i^*||}\right)}{n(\{j:\hat{y}_j=\hat{y}\})} : \hat{y}\in\mathbf{E}\right\}\right) \quad (3)$$

4.2 Feature Vector Design

The feature vector $\vec{x}_i$ for a single window i of length N is calculated from the sampled waveforms that arrive at the listener's left and right eardrums, $\vec{l}_i \in \mathbb{R}^N$ and $\vec{r}_i \in \mathbb{R}^N$. It is composed of three parts, the broadband ILD $x_{\text{BB,i}}$, spectral ILD features $\vec{X}_i$, and spectral phase differences $\vec{\phi}_i$ (Eq. 4). This last component effectively encodes the ITD due to the relationship of phase in the Fourier domain to a shift in the time domain.

$$\vec{x}_i = \left[\ x_{\text{BB},i},\ \vec{X}_i,\ \text{Re}\left(\vec{\phi}_i\right),\ \text{Im}\left(\vec{\phi}_i\right)\ \right] \quad (4)$$

The broadband ILD $x_{\text{BB},i}$ is calculated as twice the difference between the left and right channels' log root-mean-square (RMS) value (Eq.5), which is directly proportional to the difference in power level, expressed in decibels.

$$x_{\text{BB}} = \log(\vec{l}_i\cdot\vec{l}_i) - \log(\vec{r}_i\cdot\vec{r}_i) \quad (5)$$

To calculate the spectral features, the DFT of the left and right waveforms are calculated, $\vec{L}_i \in \mathbb{C}^N$ and $\vec{R}_i \in \mathbb{C}^N$. To ensure that the DFT can be calculated efficiently, the window length N is restricted to be a power of two, and ranges from 1024 to 8192 samples. This corresponds to a duration of 23-186 ms. Humans are capable of performing localization on sounds as brief as 250 ms, so a longer window duration than this would be unsupported [21].

Each of the spectral portions of the feature vector are represented by a number of frequency bins corresponding to the Mel scale. There are several competing definitions of the Mel scale [22], but they all attempt to maintain the property that a constant shift along the scale represents a constant perceived pitch change. This paper uses the definition provided by the Librosa Python package [17], where M_k is a filter bank matrix which maps a DFT onto k perceptually-uniform bins. These bins span the 4-16 kHz range known to be significant to the sound localization task [10]. The size of each of filter bank is drawn from a log-uniform distribution that ranges from 8 to 256 bins. Because the Mel scale is logarithmic, the lower-frequency bins will be calculated from fewer DFT samples than higher-frequency bins; increasing the bin count above 256 can result in these lower-frequency bins having a lower bandwidth than the DFT resolution.

The spectral ILD is calculated from the DFT similarly to the broadband ILD except that the contribution of each frequency component is weighted according to the Mel-filterbank M_n (Eq. 6), where n is a hyperparameter representing the filter bank size. Here, the single vertical bars represent the component-wise absolute value and the superscript 2 represents a component-wise squaring operation. This will result in a different scaling factor than the broadband ILD calculation; this is corrected during the MLP training process.

$$\vec{X}_i = \log(|\vec{L}_i|^2 M_n) - \log(|\vec{R}_i|^2 M_n) \quad (6)$$

To calculate the phase portion of the feature vector $\vec{\phi}_i$, a vector $\vec{\phi}_i^*$ of DFT-domain phase differences is first calculated (Eq. 7); each component of this vector is a unit-length complex number. [6] These are then weighted by the coefficients in the Mel-filterbank M_m, where m is a hyperparameter of the model and re-normalized to unit length (Eq. 8). To avoid problems with discontinuities in an angular representation, the real and complex parts of this vector are included in the feature set separately.

$$\forall k \in [1,N] : \vec{\phi}_i^*[k] = \frac{\vec{L}_i[k]\ \left|\vec{R}_i[k]\right|}{\vec{R}_i[k]\ \left|\vec{L}_i[k]\right|} \quad (7)$$

$$\vec{\phi}_i = \frac{\vec{\phi}_i^* M_m}{|\vec{\phi}_i^* M_m|} \quad (8)$$

4.3 Hyperparameter Search

Several hyperparameters are described in Secs. 4.1 and 4.2, which fall into two broad categories. The selection of features can be adjusted by changing the window size N and by choosing how many bins are used in each of the two filter banks, m and n. The network training can be adjusted by changing the activation function f_{hidden}, the regularization coefficient α, or the Adam parameters β_1 and β_2.

Appropriate values for these hyperparameters are obtained using the Optuna optimization framework [19]. [7] Each trial trains models for 3 different HRTFs with the hyperparameters suggested by Optuna, and submits the most pessimistic result as the overall trial result. The study consists of 361 trials with two optimization targets, the model accuracy A and an estimate of the required training effort. This estimate is

[6] The notation $\vec{v}[k]$ here refers to the k-th component of $\vec{v}$.

[7] The source code and hyperparameter search results are available at `https://2-71828.com/smc22/SMC22_files.zip`

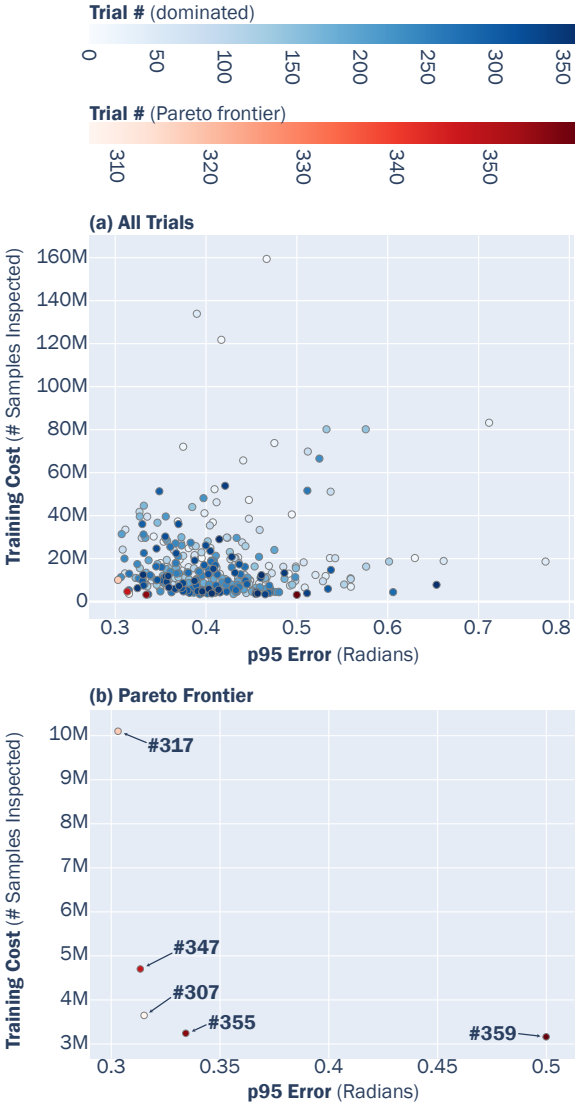


Figure 3. Trial results from Optuna study.

the product of the size of the training set $n(\mathbf{T})$ and the number of epochs processed during training. Notably absent from the estimate is any consideration of the computational cost of each training iteration, such as the differing costs of calculating gradients for the varying activation functions.

Figure 3 summarizes the results of the Optuna study. Plot (a) shows a marker for every trial in the study, and plot (b) show only those trials on the Pareto frontier, i.e. those that are better in some sense than any other trial. On plot (a), the trials on the Pareto frontier [23] are indicated in red. The color saturation indicates the trial order; later trials are shown with a more saturated color. In both plots, the accuracy score A appears on the horizontal axis and the training cost estimate on the vertical. The parameters and optimization values for the five of the trials on the Pareto frontier are reported in Table 1.

The clustering of high saturation points, representing later trials, towards the bottom-left portion of Fig. 3a indicates that the Optuna algorithm has successfully identified some

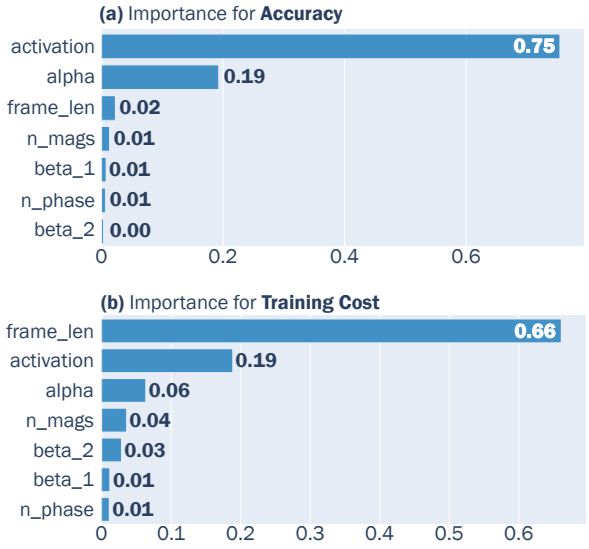


Figure 4. Hyperparameter relative importance.

properties of the hyperparameter space that promote lower training times and higher accuracy. Figure 4 shows Optuna's estimate of the relative importance of the various hyperparameters to each of the optimization targets. Though the order is different, the most important three parameters are the same for both targets: the activation function f_{hidden}, L_2 penalty coefficient α, and window size N.

The accuracy (Fig. 4a) is primarily affected by the choice of activation function f_{hidden} and the L_2 regularization coefficient α. Of the three options evaluated for the activation function, only the hyperbolic tangent appears on the Pareto frontier; this is likely due to the trigonometric character of the underlying problem. The search space for the L_2 coefficient α covered the range $[10^{-5}, 10^{-1}]$ but $\alpha \leq 5.57\times10^{-4}$ on the Pareto frontier, indicating that the upper portion of the search range is unfruitful in this application.

In contrast to the accuracy score, the most important factor in training cost is the audio window size N. Four of the five trials on the Pareto frontier feature the maximum window size of 8192 samples. This maximizes the frequency resolution of the DFT, but also minimizes the number of windows produced for each audio sample. This, in turn, minimizes the size of the MLP training set $\mathbf{T}$. There is a strong negative correlation (-0.62) between $\log_2 N$ and the training cost, and a weak positive correlation (0.16) between $\log_2 N$ and the model's error: Reducing the window size N has the potential of slightly improving accuracy at a large cost in training effort, as can be seen in the results of trial #317.

5. RESULTS AND ANALYSIS

Optuna trial #347 was selected as having the best tradeoff between accuracy and training cost. The parameters from this trial were used to train models for all 48 HRTF records in the ITF database; the results are plotted in Figure 5. With the exception of MRT02, all of the HRTFs form a dense cluster. The accuracy scores for this cluster, which represent the 95th percentile error for each individual model, have

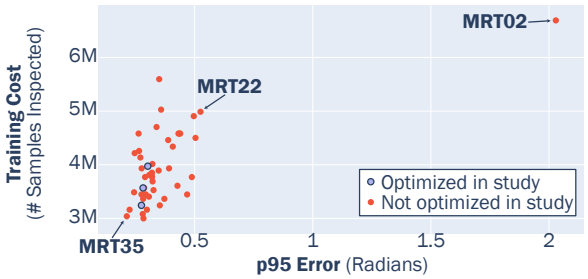


Figure 5. Performance of trained models.

a mean of 19.3° and a standard deviation of 4.45°. As the models used for the parameter search lie in the center of this cluster, there is no evidence that the search over-optimized for those models specifically.

Three of these models were selected for further analysis: the most accurate (MRT35), the least accurate within the main cluster (MRT22), and the outlier (MRT02). Their individual results are plotted in figure 6. The polar plots show the directions that were selected for evaluation and their corresponding errors. The emitter azimuth is plotted on the angular axis, with the listener facing the top of the page. The elevation angle is plotted on the radial axis: The center point represents a vertical alignment and the outer edge represents the horizontal. In each case, the sampling of test vectors is fairly uniform with a bias towards high elevation angles. This bias is inherent in the sampling method used to originally collect the ITA-HRTF data: The same number of azimuth angles were sampled for every elevation angle, so the subtended angle between samples gets denser as the elevation angle increases.

The bottom row of figure 6 shows a kernel-density estimate of the error distribution. In each case, this resembles a log-normal distribution. The maximum likelihood log-normal distribution for each of the 48 trained models predicts the observed quartiles within 14% in all cases. The median estimation error is 3.6% for the lower quartile and 4.3% for the upper quartile.

The color of each marker represents the mean error of sound samples from the indicated direction, where blue points represent a correct prediction and red points indicate a significant error. The two models from the main cluster both exhibit a few randomly-placed outliers and some general regions that have a relatively lower accuracy: MRT35 generally performs well on sounds coming from ahead or above, but performance degrades as the emitter moves to the side. Sounds from very low elevation angles in the left-rear quadrant are particularly troublesome. MRT22 has a reasonable accuracy for sounds that come from above and ahead, but significantly worse performance for all sources below the horizontal.

The outlier, MRT02, still shows good performance for sounds that appear from directly overhead, where the sampling density is highest. Aside from that, the errors appear to be quite uniformly distributed. Where records 22 and 35 sampled elevation angles with a 5° period, the elevation angles in record 2 are sampled with a 10° period. This indicates that the hyperparameter configuration is strongly dependent on the distribution of sample directions in the HRTF.

6. CONCLUSIONS

The hyperparameter search described here takes 3 days to complete on a 6-core Intel i7-10750H processor. This high computational cost severely limits the dimensionality of the hyperparameter search space, rendering larger studies intractable, such as exploring different network topologies. The authors plan to replicate this model in a high-performance computing (HPC) environment to enable such larger studies.

Overall, the developed model performs comparably to humans. When presented with a short-duration sound sample, humans are able to locate a sound source within a p95 confidence interval of (-20.2°, 21.6°) azimuth and (-23.5°, 32.2°) elevation [21]. The average model presented here has a 95^{th} percentile error of 19.3°. Excluding the one outlier, the worst model observed has a 95^{th} percentile error of 30.1°.

The derived hyperparameters show a good resilience to HRTF contents, as long as the distribution of sampled HRTF directions matches the model HRTFs used for the hyperparameter search. Coupled with the fact the features have been derived from characteristics known to be important in the human localization process, this provides strong evidence that this is a good computational model for human localization, suitable for use in evaluating spatial audio systems.

Acknowledgments

This work was performed in the Center of Excellence (CoE) Research on AI- and Simulation-Based Engineering at Exascale (RAISE) receiving funding from EU's Horizon 2020 Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733.

The work received support from the NordForsk's Nordic Sound and Music Computing Network (NordicSMC), project number 86892.

Icelandic HPC Competence Center is funded by the EuroCC project that has received funding from the European HPC Joint Undertaking (JU) under grant agreement No 951732. The JU receives support from the EU's Horizon 2020 research and innovation programme.

7. REFERENCES

[1] R. Pelzer, M. Dinakaran *et al.*, "Head-related transfer function recommendation based on perceptual similarities and anthropometric features," *The Journal of the Acoustical Society of America*, vol. 148, no. 6, pp. 3809–3817, 2020. [Online]. Available: https://doi.org/10.1121/10.0002884

[2] M. Dellepiane, N. Pietroni *et al.*, "Reconstructing head models from photographs for individualized 3d-audio processing," *Computer Graphics Forum*, vol. 27, no. 7, pp. 1719–1727, 2008. [Online].

	#307	**#317**	**#347**	**#355**	**#359**
Feature Set Parameters					
Audio samples per window N	8192	2048	8192	8192	8192
Number of spectral ILD bands n	60	177	84	164	242
Number of spectral phase difference bands m	34	60	192	34	34
Training Hyperparameters					
Node activation function f_{hidden}	tanh	tanh	tanh	tanh	tanh
L_2 normalization coefficient α	9.86×10^{-5}	5.57×10^{-4}	1.67×10^{-4}	3.16×10^{-5}	1.04×10^{-4}
Adam coefficient β_1	1.37×10^{-2}	8.88×10^{-2}	6.85×10^{-2}	4.62×10^{-2}	6.20×10^{-2}
Adam coefficient β_2	4.43×10^{-3}	6.86×10^{-3}	8.60×10^{-3}	3.40×10^{-3}	9.70×10^{-3}
Optimization Targets					
95^{th} percentile error (radians)	0.315	0.303	0.313	0.334	0.500
Training cost (# samples inspected)	3.65×10^{-6}	1.01×10^{-7}	4.70×10^{-6}	3.24×10^{-6}	3.16×10^{-6}

Table 1. Hyperparameter and optimization values of the Pareto frontier.

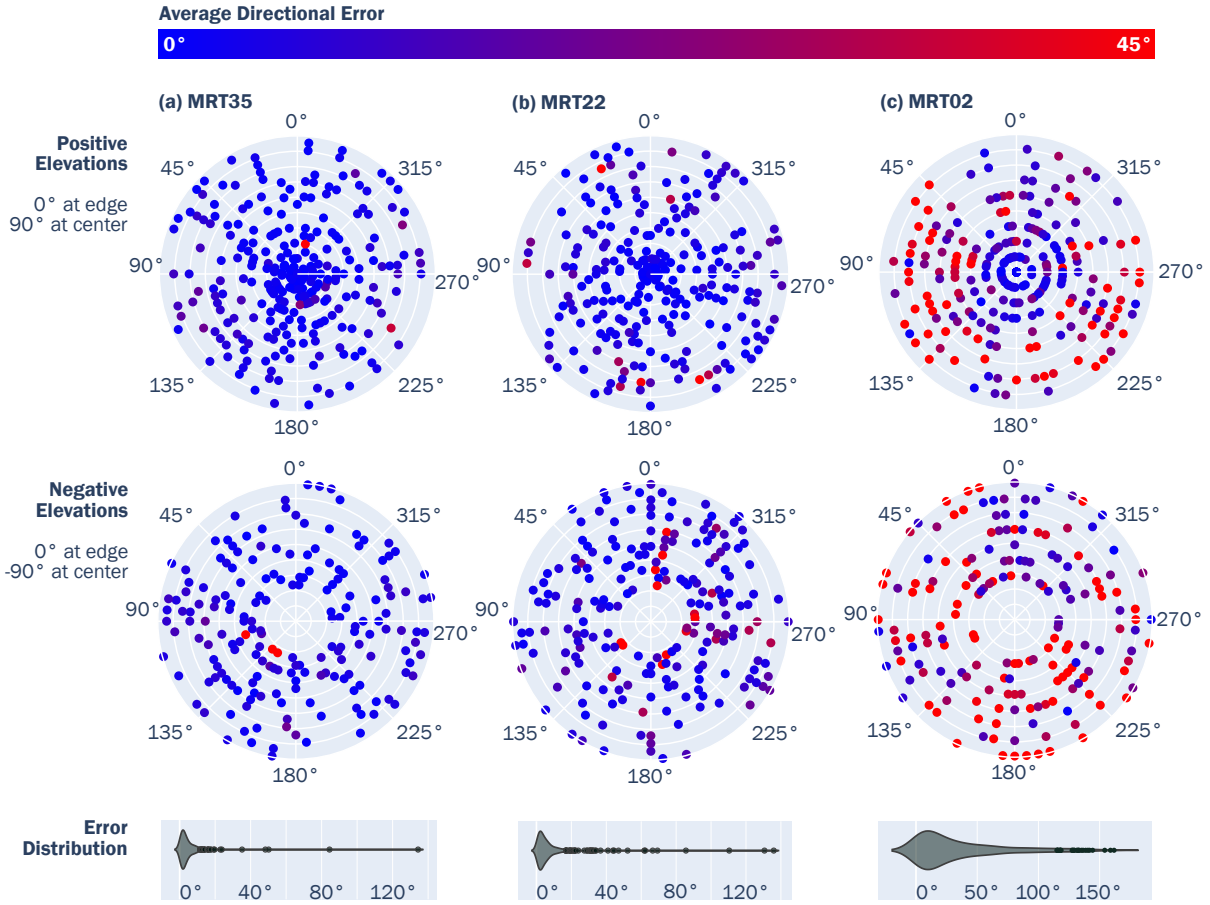


Figure 6. Performance detail of selected models.

Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2008.01316.x

[3] C. Jenny and C. Reuter, "Usability of individualized head-related transfer functions in virtual reality: Empirical study with perceptual attributes in sagittal plane sound localization," *JMIR Serious Games*, vol. 8, no. 3, p. e17576, Sep 2020. [Online]. Available: http://games.jmir.org/2020/3/e17576/

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[5] R. Bomhardt, M. de la Fuente Klein, and J. Fels, "A high-resolution head-related transfer function and three-dimensional ear model database," *Proceedings of Meetings on Acoustics*, vol. 29, no. 1, p. 050002, 2016. [Online]. Available: https://asa.scitation.org/doi/abs/10.1121/2.0000467

[6] P. M. Hofman, J. G. Van Riswick, and A. J. Van Opstal, "Relearning sound localization with new ears," *Nature Neuroscience*, vol. 1, no. 5, pp. 417–421, Sep 1998. [Online]. Available: https://doi.org/10.1038/1633

[7] H. Gray and W. H. Lewis, *Anatomy of the Human Body*, 20th ed. Philadelphia: Lea & Febiger., 1918, [Online]. Available: Bartleby.com, 2000. www.bartleby.com/107/.

[8] H.-J. Kim, D.-G. Jee *et al.*, "The real-time implementation of 3d sound system using dsp," in *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, vol. 7, 2004, pp. 4798–4800 Vol. 7.

[9] J. Sodnik and S. Tomazic, "Directional information in head related transfer functions," vol. A, 12 2004, pp. 100 – 103 Vol. 2.

[10] J. Hebrank and D. Wright, "Spectral cues used in the localization of sound sources on the median plane," *The Journal of the Acoustical Society of America*, vol. 56, no. 6, pp. 1829–1834, 1974. [Online]. Available: https://doi.org/10.1121/1.1903520

[11] R. Baumgartner, P. Majdak, and B. Laback, "Modeling sound-source localization in sagittal planes for human listeners," *The Journal of the Acoustical Society of America*, vol. 136, no. 2, p. 791—802, August 2014. [Online]. Available: https://europepmc.org/articles/PMC4582445

[12] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 1436–1445.

[13] T. Kluyver, B. Ragan-Kelley *et al.*, "Jupyter notebooks - a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Scmidt, Eds. Netherlands: IOS Press, 2016, pp. 87–90. [Online]. Available: https://eprints.soton.ac.uk/403913/

[14] *AES Standard for file exchange - Spatial acoustic data format*, AES69-2020, Audio Engineering Society, Inc., 2020.

[15] C. R. Harris, K. J. Millman *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[16] W. Cochran, J. Cooley *et al.*, "What is the fast fourier transform?" *Proceedings of the IEEE*, vol. 55, no. 10, pp. 1664–1674, 1967.

[17] B. McFee, C. Raffel *et al.*, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8. Citeseer, 2015, pp. 18–25.

[18] F. Pedregosa, G. Varoquaux *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] T. Akiba, S. Sano *et al.*, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, New York, USA, 2019, p. 2623–2631. [Online]. Available: https://doi.org/10.1145/3292500.3330701

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[21] A. Bronkhorst, "Localization of real and virtual sound sources," *Journal of the Acoustical Society of America*, vol. 98, 11 1995.

[22] F. Zheng, G. Zhang, and Z. Song, "Comparison of different implementations of mfcc," *Journal of Computer science and Technology*, vol. 16, no. 6, pp. 582–589, 2001.

[23] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *J. ACM*, vol. 22, no. 4, p. 469–476, oct 1975. [Online]. Available: https://doi.org/10.1145/321906.321910

Comparing Sonification Strategies Applied to Musical and Non-Musical Signals for Auditory Guidance Purposes

Prithvi Ravi Kantan
Aalborg University, Copenhagen
prka@create.aau.dk

ABSTRACT

Researchers have increasingly explored the potential of interactive auditory guidance in navigation and spatial orientation tasks. Despite laboratory promise, the adoption of these applications in real-life remains limited, partly due to the lack of aesthetic considerations in the design of auditory guidance stimuli, leading to auditory fatigue and low user acceptance. Although music has been suggested as a solution and tested in motor learning and rehabilitation, there is a lack of empirical research comparing its guidance efficacy with traditional nonmusical designs. Through a one-dimensional guidance task with 18 participants, the present study compared an array of novel musical strategies with nonmusical strategies based on the same auditory perceptual dimensions. It was observed that the musical strategies elicited higher user experience ratings while affording comparable performance (error, acquisition time, overshoots) to the nonmusical strategies. There were also performance differences based on the auditory dimensions manipulated by the strategies (e.g. pitch, loudness). There is thus preliminary evidence that music warrants more serious consideration as a means to address the issues of pleasantness and user preference in auditory guidance.

1. INTRODUCTION

Aided by technological advances in the sound and music computing field, digital audio applications have been increasingly explored as assistive tools in otherwise non-audio domains [1]. One such tool is *auditory guidance (AG)*, wherein a human user is interactively guided through a real-life task by an auditory stimulus. Although most conventional guidance systems communicate with human users through the visual modality (e.g. maps, medical displays, traffic signals, etc.), there are certain situations where the auditory modality may be more appropriate. This could be due to the visual modality being absent due to blindness or preoccupied with other tasks, coupled with the fact that the auditory perceptual channel is typically open and capable of processing parallel streams of information [1,2]. The utility of AG has been researched for indoor navigation [3], locomotion guidance [4], pedestrian navigation [5], spatial orientation tasks [6], precision tasks [7], and aircraft control [8] with promising results that demonstrate performance benefits from AG.

An AG signal may be verbal (such as GPS-based navigation instructions on Google Maps) or nonverbal (commonly seen in parking assistance systems), and continuous or discrete. Regardless of the specific application, AG systems work by computing the instantaneous state of the user, comparing it to the state value corresponding to task completion (target state), and using their difference (in the form of a normalized distance) as a control variable for real-time audio synthesis or processing [9]. The output is an audio signal that varies based on this distance in such a way that it guides the user towards the target. This can be seen as a case of interactive sonification, where humans interact with a system that converts data to sound [10]. The sonification topology may be 1D (only one state variable is sonified, e.g. angular distance during indoor navigation [3]) or multidimensional (e.g. two sonified Cartesian coordinates to guide 2D spatial orientation tasks [6]). This can be seen as real-time *parameter mapping sonification* [11], where discrete variables are mapped to separate auditory perceptual dimensions (e.g. pitch, loudness, tempo, timbre), ideally selected based on the task needs [12] as well as the quantity being represented [13]. Many algorithms can be used, from simple methods like subtractive/FM synthesis to more complex ones such as physical models and granular synthesis, or even DSP applied to pre-recorded audio signals such as music tracks [11]. Design rationales may be metaphorical, psychoacoustic, spatialization-based, or abstract [4, 6, 7, 9].

Despite their potential, AG applications enjoy only limited adoption and acceptance in real-life contexts at present [2, 9], an issue that also plagues the field of auditory display and sonification at large [14–16]. Researchers have attributed this to several possible causes that ultimately relate to the lack of interdisciplinary collaboration in the research field [1, 2]; these range from design philosophies and aesthetics to empirical evaluation practices. Specifically, the sonification design process has tended to lack formalization [9], leading to the prevalence of ad-hoc design practices and trial-and-error based research [1]. Design approaches have also tended to be simple and devoid of psychoacoustically or psychologically driven motivations for sound design decisions [2]. Moreover, there is a shortage of studies comparing multiple sonification strategies [2, 9]. Even those that have done so [12] have stuck to simple

Copyright: © 2022 Prithvi Ravi Kantan et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

sound designs and not explored the impact of more aesthetic approaches (e.g. music) on guidance task outcomes.

Here, my objective was to experimentally compare nonmusical and musical AG strategies based on common auditory perceptual dimensions in terms of task performance and user experience in a 1D guidance context. The following sections present related research as well as the design and implementation of the present sonification strategies, which were finally evaluated in a 1D task similar to [12] with 18 normal-hearing participants.

2. RELATED RESEARCH

2.1 Strategies Used in Auditory Guidance

Sonification strategies in AG have been classified based on their perceptual characteristics. Parseihian et al. (2015) [9] classified strategies depending on the presence of a sonic reference denoting the target state. They defined three types - *without reference* (e.g. sine pitch), *with reference* (e.g. synchronicity, harmonicity), and *reference-and-zoom* (e.g. multiband frequency modulation). At a lower level, strategies can also be classified on the basis of the auditory properties they manipulate. Pitch and loudness are commonly used dimensions [9], but more complex psychoacoustic dimensions such as chroma, beating, and roughness have successfully been tested as well [6].

The appropriate sonification strategy for an AG task depends upon the spatiotemporal constraints of the task, and past studies have typically used evaluation metrics such as target acquisition error, time, and overshoots [2,3,6,12]. In short, most studies have found AG to be feasible and beneficial [2–6]. Many have compared the efficacies of AG and visual guidance, mostly finding that visual guidance is faster and more accurate [5, 6], although AG outperforms it in some cases [7]. The former is not surprising, considering that vision is far more precise, and subject to far less individual variability than audition [15]. But even purely within AG, studies have tended to lack rigorous empirical comparisons of different strategy types [2]. A notable exception is [12], where multiple strategies were compared in the context of a 1D target-finding task. The authors found systematic differences in task performance depending on the nature of the sonification strategies (auditory dimension, presence of sonic reference) and the task goal (speed, accuracy, or overshoot minimization). They did not, however, investigate the implications of sonic aesthetics in the context of their designed strategies [12].

2.2 The Issue of Aesthetics

Aesthetics, which binds together aspects such as intrusiveness, listener fatigue, annoyance, and comprehensibility of a sonification [17], can impact the utility, usability, and eventual acceptance of an application. Human-computer interaction research has shown that aesthetic values are an important determinant of user preferences [18], and similar considerations apply to sonification design [2, 9, 19–21]. A lack of aesthetically pleasing and preference-tailored sonification designs may contribute to ongoing problems with acceptance and adoption [2, 9, 20]. Despite the notion that the functional and aesthetic properties of auditory displays cannot be dealt with independently [17, 21], it is common to make use of simple sounds (e.g. sine waves, noise) and perceptual dimensions (qualities - e.g. pitch, loudness) [13]. This, aside from not catering to user tastes, can lead to auditory fatigue [2, 9, 19], although acceptance and fatigue have been found to be subjective [6].

To alleviate these concerns, several approaches have been proposed and tested - these are based on the premise that it is possible to apply variations of an auditory dimension to not only simple sounds, but also complex sound textures without affecting sonification performance [12]. For instance, morphological earcons [9] essentially convey information through *sonic evolution along auditory dimensions*, rather than the chosen sounds themselves; this creates a common 'semantic language'. It is thus possible to satisfy individual preferences and allow seamless switching between various sound palettes without significant changes in cognitive load or learning time [9]. Indeed, many complex 'sound palettes' have successfully been tested, such as physically modeled instruments [7] and natural/electronic soundscapes [9], but we restrict our focus to the one type of organized sound with universal appeal - music.

2.3 Music as an AG Medium

Although musical sonification has been explored to a considerable extent in the domain of motor learning and movement rehabilitation for its ability to motivate and modify movement [22–24], music as a *guidance* medium has been less common. This may be due to the reluctance of designers to use a complex and dynamic signal for fear of the seeming ambiguity introduced to data when it is represented using aesthetic approaches [20]. But while it is indeed true that musical sonification designs have a less direct data-sound relationship (indexicality [17]) than more traditional designs, there is a little empirical evidence that musical AG is prohibitively imprecise as a result. In fact, experiments have shown that adaptive DSP (panning, tempo changes) applied to user-selected music can serve as an effective guidance tool [8,23], and even simple scale melodies have been shown to be effective AG tools [3].

If an AG signal must be continuously audible (as asserted in several works [6, 8]), then music may be a viable means to address the problems of aesthetics and personal preference. The goal of the present study was to compare musical and nonmusical strategies in terms of objective outcomes (e.g. error, speed) and elicited user experience. This could serve as empirical evidence to justify the incorporation of layered musical material in AG design, providing a better understanding of *how* musical materials can be manipulated so as to generate effective AG. In this study, the term 'musical sonification' refers to the use of audio DSP processing on continuous music signals to convey guidance information.

3. MATERIALS AND METHODS

For the present study, two sets of AG strategies were tested - *musical* and *nonmusical*. For the nonmusical set, a subset

of the strategies used in [12] were reproduced. The novel musical strategies were designed such that each nonmusical strategy had a musical *correlate* based on the same auditory dimension. Whilst the nonmusical strategy manipulated a stationary signal (sine tone, noise, beep train), its musical counterpart manipulated the characteristics of a synthesized multitrack musical arrangement (described next). The *reference-and-zoom* strategies in [12] were excluded due to the complexity of adapting them for a layered musical signal. The experimental evaluation was a simplified recreation of the 1D target-finding task in [12], where an unspecified point on a line (target) had to be located with the help of AG. To generate the present strategies, real-time music sequencing was carried out in a JUCE-based [1] C++ program, and FAUST [2] was used for audio synthesis, processing, and mixing.

3.1 Music Generation

The sequencing functionality in JUCE was written to provide note (triggering, pitch, velocity) information to the synthesis engine at sixteenth note intervals at 120 BPM. The synthesis engine in FAUST generated eight tracks corresponding to pop/electronic instruments. These included percussive and melodic instruments in several frequency registers, enabling syncopation and true polyphony in a 4/4 time signature. The synthesis methods were based on simple waveforms (e.g. square, sawtooth, noise) with envelope-controlled filters, as well as FM with simple attack-release envelopes. Parametric equalizers and dynamic range compressors are applied so as to attain the desired balance prior to stereo mixdown. The source code can be found in the attached web repository [3]. Melodic and harmonic patterns corresponding to four popular songs [4] were encoded into the sequencer for the purpose of this study.

3.2 AG Strategy Design

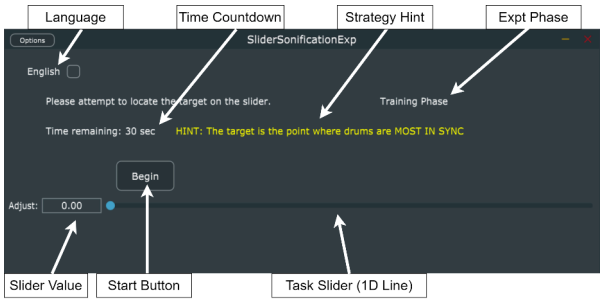


Figure 1. The JUCE-built interface used to carry out 1D AG tasks using the different strategies. The mouse was used to initiate tasks and adjust the slider value (user position along the line segment). Tasks were concluded by hitting the space bar.

The strategies were designed for 1D guidance, wherein the task goal was to navigate along a line segment to an unknown *target* point using AG as shown in Fig. 1. Hence the sonified quantity was the distance x between the instantaneous position of the user along the segment and the target. x was normalized between 0 and 1, where 1 corresponded to the total length of the segment. Six auditory dimensions were considered, each with nonmusical and musical AG strategy correlates. The nonmusical strategies are only briefly described here (see [12] for the motivation behind the parameter ranges). Also, the term *melody instruments* refers henceforth to the combination of all pitched instruments (bass synth, chord synths, main melody synth). Audiovisual examples of all strategies are provided [5].

3.2.1 *Nonmusical Strategies*

- **Pitch (P):** x was positively mapped to the fundamental frequency of a pure tone through a logarithmic function that scaled the tone frequency in the 300 Hz - 3.4 kHz range.
- **Loudness (L):** x was positively mapped through a logarithmic power function to the amplitude of a sine wave at 600 Hz, such that the dynamic range corresponding to the x value range was 40 dB.
- **Brightness (Br):** x controlled the bandwidth of a white noise signal via the cutoff frequency of a 2nd order lowpass filter, such that the cutoff frequency range was 300 Hz - 3.4 kHz (maximum at $x = 0$, with logarithmic scaling).
- **Amplitude Modulation (AM):** A 200 Hz sine tone was added to a second sine tone whose frequency was controlled by x, such that their summation produced a beating/amplitude modulation effect at frequencies ranging from 10 Hz to 0 Hz (no beating at $x = 0$).
- **Synchronicity (Sy):** x was mapped to the temporal offset between two identical beep sequences at the same tempo (120 BPM), such that the beeps were perfectly synchronized when x was 0, and offset by 125 ms when x was 1.
- **Harmonicity (Hm):** x was mapped to the higher partial frequencies of a sinusoidal harmonic series, such that increasing x introduced an inharmonicity factor of up to 0.01.

3.2.2 *Musical Strategies*

- **Pitch (P):** x was mapped to the root note frequency of the melody instruments as a multiplicative factor, causing them to be transposed upward by a maximum of 60 semitones as:

$$fTonic_{final} = fTonic_{original} \cdot (1 + 5 \cdot x)$$

[1] JUCE Framework - https://juce.com/
[2] FAUST Language - https://faust.grame.fr/
[3] https://github.com/prithviKantanAAU/SMC2022_AG_Code_Data
[4] The Eagles - Hotel California, Tom Petty and the Heartbreakers - Free Fallin', Luis Fonsi - Despacito
[5] AG Strategy Demo Video: https://www.youtube.com/watch?v=RMbzq1rydrY

- **Loudness (L):** x was mapped to the gain control of the melody instruments, such that increases in x resulted in these instruments becoming softer as:

$$MelGain_{dB} = -80 \cdot x$$

- **Brightness (Br):** x controlled the cutoff frequency of a resonant low pass filter on the melody instruments such that the cutoff frequency was maximum on the target and decreased with increases in x. The filter had a Q-factor of 2.5 to highlight cutoff frequency changes.

$$fc_{Mel}(Hz) = 12000 - 11800 \cdot x$$

- **Amplitude Modulation (AM):** x was mapped to the depth control of a tremolo effect (amplitude modulation) that processed the melody instruments. The tremolo effect consisted of a sine *LFO (low frequency oscillator)* at the musical beat frequency. Increasing x increased the effect intensity as:

$$AMProd = LFO \cdot MelMix_{pre}$$

$$MelMix_{final} = MelMix_{pre} \cdot (1 - x) + AMProd \cdot x$$

- **Synchronicity (Sy):** x was mapped to temporal offset factors separately applied to all instruments (realized using delay buffers), reducing their level of musical synchronization as x increased. The delay in seconds for each instrument was calculated as:

$$d(Inst) = d_{Max}(Inst) \cdot x$$

- **Harmonicity (Hm):** x was mapped to the mix ratio control of a ring modulator effect connected to the melody instruments. The modulation frequency of the effect was configured to the tritone relative to the tonic, as this was found to create the most inharmonic result compared to other frequencies. Increasing x made the output more dissonant as:

$$RMProd = MelMix_{pre} \cdot sin(2\pi f_{mod} t)$$

$$MelMix_{final} = MelMix_{pre} \cdot (1 - x) + RMProd \cdot x$$

4. EXPERIMENTAL EVALUATION

An experiment was carried out with the purpose of directly comparing the AG strategies in terms of guidance efficacy and user experience. At the outset, I framed the following hypotheses based on past literature:

- H1: Participants will exhibit inferior performance with the musical strategies than with their nonmusical correlates, manifesting as greater error, longer acquisition time, and more overshoots per trial.
- H2: There will be differences in task performance between the auditory dimensions (e.g. brightness and synchronicity will lead to greater error than pitch and loudness).
- H3: The musical strategies will be rated as more pleasant and preferable for longer use durations than their nonmusical correlates.

4.0.1 Participants

A convenience sample of 18 participants (3 women) from Aalborg University aged 29.4 $\pm$ 6.3 years volunteered to participate in the experiment. Each of them was briefed about the purpose and length of the experiment, and informed that they could withdraw at any time. All experimental procedures conformed to the ethics code of the Declaration of Helsinki. Informed consent was obtained prior to participation, and no sensitive or confidential information was collected from the participants.

4.0.2 Experimental Setup

The experiment was conducted in a quiet room at Aalborg University. It took place on a Dell laptop loaded with the experimental interface (see Fig. 1), which provided instructions, set up the AG tasks, and recorded participant responses. An external keyboard and mouse were used to operate the software, and sound was played using Audio Technica ATH-M50X headphones.

4.0.3 Procedure

After the initial briefing, the participant was asked to complete an online musical background questionnaire to determine their *Ollen Musical Sophistication Index (OMSI)* [25]. The experiment was conducted using the interface, which randomized the presentation order of the 12 AG strategies. For each strategy, the procedure was as follows:

- Training Phase: The participant familiarized themselves with the AG strategy during this phase. An on-screen hint was first provided (see Fig. 1 for an example). Based on this, they attempted to locate a randomly determined target point using the AG strategy. There was no time limit for this, and a feedback prompt appeared when the participant was in the vicinity of the target value. When the participant felt comfortable with the strategy, they could proceed to the main testing phase.
- Testing Phase: This was identical to the training phase, except that there was a 30 second time limit, and no prompt appeared in the target vicinity. The instruction was to find the target point *'as quickly and accurately as possible within the time limit'*. Participants pressed the space bar to conclude the trial when complete, at which time the trial data were logged. Following the guidance task, the participant was given accuracy feedback and asked to rate two qualitative measures on a 7-point Likert scale - (A) how pleasant they found the guidance strategy, and (B) the duration for which they would prefer to use the AG strategy.

The above procedure was repeated for all 12 AG strategies. During both phases, the random target location was constrained to lie between 50% and 85% of the slider length so as to make the initial approach clearly audible and prevent unavoidable overshoots.

4.0.4 Data Analysis

The data were organized in MATLAB 2018b and statistically analyzed in SPSS 27.0. The outcome measures *(abbreviations in parentheses)* were:

- Absolute Error % *(Error)*
- Acquisition Time *(Time)*
- Target Overshoots *(Overshoots)*
- Rated Pleasantness *(Pleasantness)*
- Rated Duration of Preferred Use *(PrefDuration)*

These outcomes were compared both *between and within auditory dimensions*, respectively using repeated-measures (RM) ANCOVA analyses between (i) the six auditory dimensions (each with their two correlated strategies averaged) **[Hypothesis H2]**, and (ii) the 12 AG strategies, with participant OMSI score *(OMSI)* as the covariate. I then examined differences between the musical and nonmusical correlates within each auditory dimension to test **H1** and **H3**. A significance criterion $\alpha = 0.05$ was used for all analyses, and post-hoc Tukey tests were carried out when significant main effects and/or interactions were detected. In addition, Pearson correlation coefficients were computed between each pair of outcomes to investigate possible linear associations. Summary measures for the data are presented as **mean** $\pm$ **standard deviation**. The JUCE code, data logs, analysis scripts, and SPSS test outputs are available on GitHub.

5. RESULTS

We found that the participants (OMSI 252.11 $\pm$ 275.06) were able to perform the AG tasks with an overall error of **2.59** $\pm$ **1.94 %**, taking **16.17** $\pm$ **4.29 sec** to complete each task, and committing **7.16** $\pm$ **2.91 target overshoots** per task. Table 1 shows the Pearson correlation coeffients between each pair of outcomes across each of the 216 individual trials (18 participants $\times$ 12 trials), revealing several significant correlations. For the ANCOVA analyses carried out across auditory dimensions and strategies, there were no significant main effects of the covariate (OMSI), and between-subject statistics are therefore not reported.

5.1 Between Auditory Dimensions

The RM ANCOVA revealed a significant main effect of *Auditory Dimension* on *Error* (F(2.34, 37.51) = 8.926, $p < 0.001$, $\eta_p^2 = 0.258$) with post-hoc comparisons revealing significant pairwise differences among several auditory dimensions (Fig. 2 (A)). It is seen that the **Loudness** strategies elicited the least error (0.90 $\pm$ 1.36 %) and the **Synchronity** strategies elicited the greatest (5.47 $\pm$ 4.79 %).

	Er	**Tm**	**Ov**	**Ple**	**PDur**
Er	1	0.132	**-0.153**	-0.126	-0.067
Tm	-	1	**0.296**	**0.139**	**0.154**
Ov	-	-	1	0.018	-.002
Ple	-	-	-	1	**0.739**
PDur	-	-	-	-	1

Table 1. The upper triangular portion of the Pearson's correlation matrix between all outcomes across the 216 recorded trials. **Bold and underlined** values indicate significant correlations. Er = Error, Tm = Time, Ov = Overshoots, Ple = Pleasantness, PDur = PrefDuration

There was also a significant main effect on *Time* (F(5, 80) = 11.395, $p < 0.001$, $\eta_p^2 = 0.416$), with post-hoc comparisons showing several pairwise differences (Fig. 2 (B)). Participants were fastest when using the **Loudness** strategies (12.48 $\pm$ 4.38 sec), and slowest when using the **Amplitude Modulation** strategies (20.40 $\pm$ 5.64 sec). There was no main effect of *Auditory Dimension* on *Overshoots* (F(2.83, 45.38) = 1.836, p = 0.157, $\eta_p^2 = 0.103$) or *Pleasantness* (F(5, 80) = 1.895, p = 0.106, $\eta_p^2 = 0.106$). There was, however, a main effect on *PrefDuration* (F(5, 80) = 2.625, p = 0.03, $\eta_p^2 = 0.141$), with post-hoc comparisons revealing significant pairwise differences (Fig. 2 (C)). The **Loudness** strategies received the highest ratings out of 7 (3.94 $\pm$ 1.74), whereas the **Harmonicity** strategies received the lowest (2.67 $\pm$ 1.31).

5.2 Within Auditory Dimensions

The RM ANCOVA showed a significant main effect of *Strategy* on *Error* (F(3.70, 62.91) = 4.544, p = 0.003, $\eta_p^2 = 0.211$) and *Overshoots* (F(4.34, 73.78) = 2.549, p = 0.042, $\eta_p^2 = 0.13$). Post-hoc comparisons did not, however, show significant differences between correlated strategies within any auditory dimension for either outcome. There was a significant main effect on *Time* (F(11, 187) = 10.678, $p < 0.001$, $\eta_p^2 = 0.386$). Post-hoc comparisons showed that participants took significantly longer when using musical strategies than their nonmusical correlates for three of the six auditory dimensions **(Pitch, Loudness, Harmonicity)** (Fig. 3 (A)). But on the whole, the qualitative outcomes appeared to favor the musical strategies. There was a significant main effect of *Strategy* on *Pleasantness* (F(11, 187) = 9.279, $p < 0.001$, $\eta_p^2 = 0.353$), with post-hoc comparisons showing that musical strategies were rated significantly higher than their nonmusical correlates for all auditory dimensions except **Synchronicity** (Fig. 3 (B)). Lastly, there was also a significant main effect on *PrefDuration* (F(11, 187) = 4.803, $p < 0.001$, $\eta_p^2 = 0.22$). Post-hoc comparisons revealed higher ratings for the musical strategies corresponding to three auditory dimensions **(Loudness, Brightness, Harmonicity)** (Fig. 3 (C)).

6. DISCUSSION

In this study, I experimentally compared nonmusical auditory guidance strategies [12] with a set of novel musical

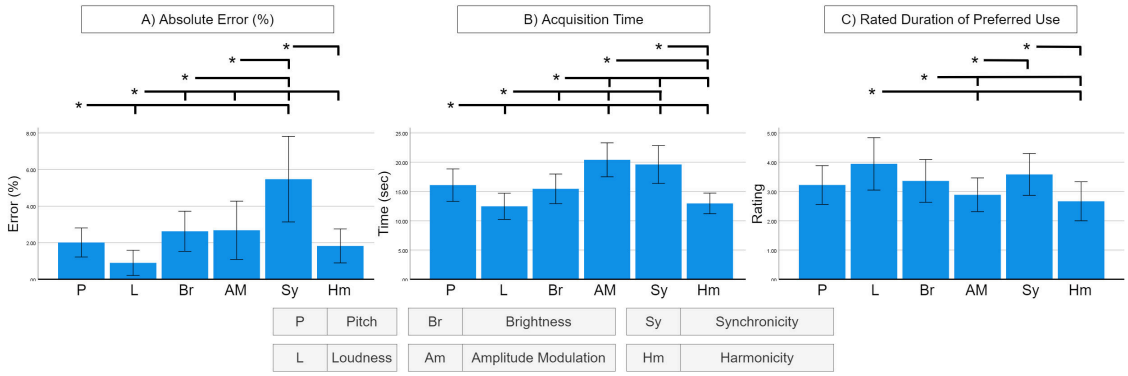


Figure 2. Selected outcomes across auditory dimensions, with constituent musical/nonmusical strategy outcomes averaged. The brackets indicate significant differences between auditory dimensions. Bar heights represent mean values, and error bars show 95% confidence intervals.

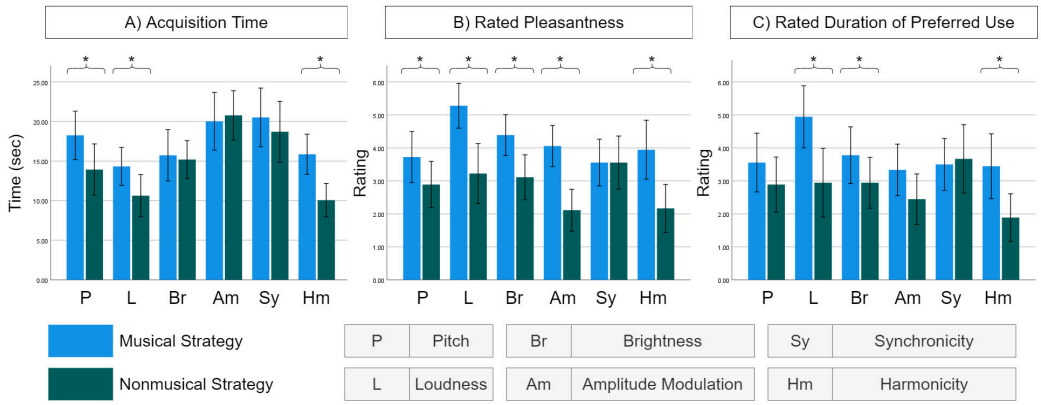


Figure 3. Selected outcomes across the individual AG strategies, with correlated musical/nonmusical strategies stacked together within the respective auditory dimension. The asterisks indicate significant differences between correlated strategies. Bar heights represent mean values, and error bars show 95% confidence intervals.

strategies based on the same auditory dimensions. They were assessed in terms of guidance efficacy and elicited user experience. Based on the known trade-off between aesthetics and precision in data representation through sound [15, 17, 20], I first hypothesized that participants would exhibit inferior performance with the musical strategies than with the nonmusical ones **(H1)**. This was partially validated in terms of acquisition time, where participants took significantly longer with musical AG for three of the six auditory dimensions. This can be attributed to heightened uncertainty when examining the pitch, loudness, and harmonicity of a complex, nonstationary signal (music) as compared to simple tones, which may have caused users to take longer to finalize their judgments.

On the other hand, there were no differences between the correlated strategies in terms of error and overshoots. This can be seen as evidence of the musical and nonmusical strategies showing comparable efficacy despite the overall background of the participants (mean OMSI 252.11, meaning 'less musically trained' [25]). But it is also likely that these results were due to the design of the experiment. Unlike in other comparable studies [6, 12] the participants received only limited practice with each AG strategy (1 round of target-finding), making it unlikely that they had reached their performance ceiling before the testing phase. It is therefore premature to comment on the peak performance allowed by the musical and nonmusical strategies.

Furthermore, the experiment did not examine the effect of different instruction types prioritizing speed, accuracy, and overshoot minimization, although past research has shown that this strongly impacts AG user behavior and performance, particularly overshoots [12]. The instruction to find the target 'quickly and accurately' may have meant that individual participants prioritized speed and accuracy differently, making it difficult to replicate the clear differences across strategies seen in [12]. Nevertheless, the significant negative correlation between overshoots and error aligns with past findings [12] in that when prioritizing accuracy (minimizing error), participants carry out small back-and-forth adjustments close to the target that manifest as overshoots. Ultimately, the relatively small sample size (18 participants) and single testing phase trial per

strategy (as opposed to three in [12]) may also have contributed to making the differences in error and overshoots between musical and nonmusical strategies harder to statistically detect.

Next, it was hypothesized that participants would perform differently with the various auditory dimensions **(H2)**. This was validated by significant differences in both error and acquisition time. Pitch, loudness, and harmonicity were the best-performing dimensions in these terms, with synchronicity performing poorest in terms of error. In the case of musical AG, this may have been due to difficulties in understanding small deviations in musical timing among several interacting rhythms, especially considering that most participants lacked musical expertise. In terms of time, AM and synchronicity took longest, most likely due to the low frequency of the AM (slow changes in volume), and the relatively large minimum time required to make rhythm-based judgments compared to e.g. pitch [26]. The overall dimensionwise trends in [12] could not be replicated, purportedly due to differences in experiment design.

The final hypothesis **(H3)** was that the musical strategies would be rated more favorably than their nonmusical counterparts in terms of pleasantness and preferred duration of use. This was largely validated by significant differences seen within almost all auditory dimensions. This is in line with past findings about users preferring algorithmic musical sonification over other designs [27], but it is worth noting that this was despite the extremely basic DSP methods used to synthesize the music. It is likely that the ratings for musical AG would be even better with user-selected pre-recorded music tracks like in [8]; most of the musical strategies (except synchronicity) can easily be adapted into DSP algorithms for recorded audio sequences.

There was also a strong correlation between *Pleasantness* and *PrefDuration*, which can be interpreted as a 'halo-effect', where the perception of aesthetic values may have strongly influenced overall user preference [18]. Although [18] found *usability* to be the other key predictor of preference, we did not directly capture any subjective measures of usability, and future studies should incorporate these so as to study the usability factors that dictate overall preferences. The user ratings were also positively correlated to acquisition time, either because the participants wilfully spent more time listening to their preferred AG strategies, or because they simply took longer to find the target using musical AG. It is worth noting the task was relatively short (30 sec) and user ratings would probably have changed with longer exposure to all AG strategies [19]. In the absence of significant correlations between the subjective ratings and error/overshoots, it also remains unclear precisely how user preferences are linked to task performance (a.k.a. guidance efficacy). It is likely that this will relationship will vary depending on the needs of the AG task at hand [21].

Limitations of the study include the limited ecological validity of the 1D AG task, static nature of the target value/state, and the lack of equivalence between how musical and nonmusical correlate strategies were generated (e.g. AM and harmonicity). It is unclear whether the findings will generalize to higher-dimensional and/or dynamic tasks, especially considering the increased cognitive load and strict need for orthogonal auditory dimensions in these situations [6, 12, 15]. In terms of gauging user experience, more established questionnaires such as the system usability scale and other HCI standards could have been used, as well as the NASA-TLX for assessing subjective cognitive load. A measure of invasiveness of the sounds was also lacking. Limitations of the experiment design such as instruction type, practice length, and total test repetitions should be addressed in future studies, which should also be conducted with a larger and more gender-balanced sample, preferably with potential users (e.g. visually impaired). Lastly, the OMSI [25] may not have been the best measure of musical ability in this context as it did not reveal any differences depending on music training, something that sonification studies have consistently found [15]. More targeted tests of rhythmic and melodic discrimination may be worth adopting in future investigations.

7. CONCLUSIONS

This study extended past research on auditory guidance by comparing the efficacy of musical strategies with previously established nonmusical ones. Aside from providing an AG performance comparison of the various auditory dimensions when used with a dynamic musical stimulus, the findings suggest that the use of music has the potential to improve user experience when using AG without introducing significant performance degradation. Although future studies should address the limitations of the present methodology and perform rigorous comparisons in real-world AG scenarios, this study provides preliminary evidence that the use of music warrants more serious consideration as a means to address present issues of pleasantness and user preference in auditory guidance design.

Acknowledgments

I would like to thank Sofia Dahl for her valuable inputs during the conception and execution of the study, as well as the individuals who took part in the experiment. This work is partially funded by by NordForsk's Nordic University Hub, Nordic Sound and Music Computing Network NordicSMC, project number 86892.

8. REFERENCES

[1] G. Kramer *et al.*, "The sonification report: Status of the field and research agenda. report prepared for the national science foundation by members of the international community for auditory display," 01 1999.

[2] D. Black, C. Hansen, A. Nabavi, R. Kikinis, and H. Hahn, "A survey of auditory display in image-guided interventions," *International journal of computer assisted radiology and surgery*, vol. 12, no. 10, pp. 1665–1676, 2017.

[3] D. Ahmetovic, U. Oh, S. Mascetti, and C. Asakawa, "Turn right: Analysis of rotation errors in turn-by-turn

navigation for individuals with visual impairments," in *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*, 2018, pp. 333–339.

[4] F. Dollack, M. Perusquía-Hernández, H. Kadone, and K. Suzuki, "Auditory locomotion guidance system for spatial localization," in *2019 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*. IEEE, 2019, pp. 1–5.

[5] A. Montuwy, B. Cahour, and A. Dommes, "Visual, auditory and haptic navigation feedbacks among older pedestrians," in *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, 2017, pp. 1–8.

[6] T. Ziemer and H. Schultheis, "Psychoacoustic auditory display for navigation: an auditory assistance system for spatial orientation tasks," *Journal on Multimodal User Interfaces*, vol. 13, no. 3, pp. 205–218, 2019.

[7] H. Roodaki, N. Navab, A. Eslami, C. Stapleton, and N. Navab, "Sonifeye: Sonification of Visual Information Using Physical Modeling Sound Synthesis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 11, pp. 2366–2371, 2017.

[8] B. D. Simpson, D. S. Brungart, R. C. Dallman, R. J. Yasky, and G. D. Romigh, "Flying by ear: Blind flight with a music-based artificial horizon," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 52, no. 1. SAGE Publications Sage CA: Los Angeles, CA, 2008, pp. 6–10.

[9] G. Parseihian, S. Ystad, M. Aramaki, and R. Kronland-Martinet, "The process of sonification design for guidance tasks," *J Mob Med*, vol. 9, no. 2, 2015.

[10] A. Hunt and T. Hermann, "Interactive sonification," in *The Sonification Handbook*, T. Hermann, A. Hunt, and J. G. Neuhoff, Eds. Logos Verlag Berlin, 2011.

[11] T. Hermann, A. Hunt, and J. G. Neuhoff, *The Sonification Handbook*. Logos Verlag Berlin, 2011.

[12] G. Parseihian, C. Gondre, M. Aramaki, S. Ystad, and R. Kronland-Martinet, "Comparison and evaluation of sonification strategies for guidance tasks," *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 674–686, 2016.

[13] G. Dubus and R. Bresin, "A Systematic Review of Mapping Strategies for the Sonification of Physical Quantities," *PloS one*, vol. 8, no. 12, p. e82491, 2013.

[14] M. Ballora, "Sonification, science and popular music: In search of the 'wow'," *Organised Sound*, vol. 19, no. 1, pp. 30–40, 2014.

[15] J. G. Neuhoff, "Is Sonification Doomed to Fail?" in *Proceedings of the 25th International Conference on Auditory Display (ICAD 2019)*. Newcastle upon Tyne: Department of Computer and Information Sciences, Northumbria University, Jun. 2019, pp. 327–330.

[16] A. Supper, "Sound Information: Sonification in the Age of Complex Data and Digital Audio," *Information & Culture*, vol. 50, no. 4, pp. 441–464, 2015.

[17] S. Barrass and P. Vickers, "Sonification design and aesthetics," in *The Sonification Handbook*, T. Hermann, A. Hunt, and J. G. Neuhoff, Eds. Logos Verlag Berlin, 2011.

[18] A. De Angeli, A. Sutcliffe, and J. Hartmann, "Interaction, usability and aesthetics: what influences users' preferences?" in *Proceedings of the 6th conference on Designing Interactive systems*, 01 2006.

[19] K. Vogt, V. Goudarzi, and R. Parncutt, "Empirical aesthetic evaluation of sonifications," in *Proceedings of the 19th International Conference on Auditory Display (ICAD 2013)*, 2013.

[20] S. D. H. Cornejo, "Towards Ecological, Embodied and User-Centric Design in Auditory Display," in *Proceedings of the 24th International Conference on Auditory Display - ICAD 2018*. Houghton, Michigan: The International Community for Auditory Display, Jun. 2018, pp. 191–196.

[21] G. Leplaitre and I. McGregor, "How to tackle auditory interface aesthetics? discussion and case study," in *Proceedings of ICAD 04-Tenth Meeting of the International Conference on Auditory Display, Sydney, Australia, July 6-9, 2004*, 2004.

[22] P.-J. Maes, J. Buhmann, and M. Leman, "3mo: A Model for Music-Based Biofeedback," *Frontiers in Neuroscience*, vol. 1, 12 2016.

[23] I. Bergstrom, S. Seinfeld, J. Arroyo Palacios, M. Slater, and M. Sanchez-Vives, "Using music as a signal for biofeedback." *International journal of psychophysiology : official journal of the International Organization of Psychophysiology*, vol. 93, 04 2013.

[24] P. Kantan, E. G. Spaich, and S. Dahl, "A technical framework for musical biofeedback in stroke rehabilitation," *IEEE Transactions on Human-Machine Systems*, 2022.

[25] J. E. Ollen, "A criterion-related validity test of selected indicators of musical sophistication using expert ratings," Ph.D. dissertation, The Ohio State University, 2006.

[26] I.-H. Hsieh and K. Saberi, "Temporal integration in absolute identification of musical pitch," *Hearing Research*, vol. 233, no. 1-2, pp. 108–116, 2007.

[27] S. Barrass, N. Schaffert, and T. Barrass, "Probing preferences between six designs of interactive sonifications for recreational sports, health and fitness," *Proceedings of ISon*, pp. 23–29, 2010.

Similarity of Nearest-Neighbor Query Results in Deep Latent Spaces

Philip Tovstogan **Xavier Serra** **Dmitry Bogdanov**
Music Technology Group, Universitat Pompeu Fabra
first.last@upf.edu

ABSTRACT

Music recommendation systems are commonly used for personalized recommendations. However, there are cases where due to privacy concerns or design decisions, there is no user information nor collaborative filtering data available. In those cases, it is possible to use content-based similarity spaces to retrieve the most similar tracks to be recommended based on the reference track. In this paper, we compare the latent spaces extracted from state-of-the-art autotagging models in terms of the similarity between lists of retrieved nearest neighbors. We additionally study item factors from collaborative-filtering data as a reference. We provide insights into how much the choice of the architecture, training dataset, or model layer (output vs. penultimate) as well as a projection of the latent space onto 2D changes the list of retrieved nearest neighbors. We release the dataset of 9 content-based and 3 collaborative-filtering latent representations of 29 275 tracks from Jamendo that we use for the evaluation. Moreover, we perform an online user experiment to compare the perceived track-to-track similarity of the selected evaluated latent spaces. The results show that content-based spaces show better results in our scenario, particularly embeddings from penultimate layers of auto-tagging architectures.

1. INTRODUCTION

In the age of prevalent music streaming, music recommendation systems are currently one of the primary ways for people to listen and find music. While collaborative filtering (CF) approaches are still within the state-of-the-art for personalized music recommendation, pure CF falls short at the cold-start problem and non-personalized recommendations. The content-based (CB) approaches can provide recommendations and suggestions based on item-to-item similarity without CF data, and they are commonly used together with CF in modern recommendation systems [1] to solve the cold-start problem. However, when there is no CF data available due to design decisions or privacy concerns, CB approaches are the only ones that can provide recommendations.

In the domain of music, there are different modalities to the content that can be used for CB approaches. Apart from the audio signal, data that can be used is metadata, user-defined tags, reviews, etc. In this paper, we will focus on audio and current state-of-the-art auto-tagging models. We are interested in how consistent are the latent spaces extracted by auto-tagging models between each other, particularly concerning the choice of the training dataset, architecture, or the layer of the network. These insights can show which variable contributes to the most dissimilar results, which can inform practical decisions on the prioritization of models for A/B testing in an industry scenario with limited resources.

Latent similarity spaces are also quite extensively used in music visualization interfaces [2], where such similarity spaces represent music on a 2D plane or 3D space and facilitate exploration, discovery, and re-discovery of music. The latent spaces usually are high-dimensional, and part of the information is lost by performing the projection. We are interested to see how well the commonly used projection methodologies represent and transform similarity space, and how much of the nearest neighbors' information is preserved.

Furthermore, we investigate how CB approaches compare to CF approaches in a user-less scenario, with CF factors representing a latent similarity space. The motivation is to see if different CB approaches capture more or less of the information that comes from user interactions in CF systems, thus resulting in more or less similar nearest neighbor results.

2. RELATED WORK

Music similarity is a widely researched topic in music information retrieval. In MIREX (Music Information Retrieval Evaluation eXchange) the task of music similarity has been active until 2015, as eventually, the performances of the submitted systems have reached the glass ceiling stemming from evaluation being subjective and limited inter-rater agreement [3]. The music similarity is quite subjective as humans use different dimensions for assessing similarity: genre, moods, tempo, instrumentation, etc. Recent work investigates the importance of inter- and intra-rater agreement in the context of music similarity and recommendation [4] that questions the validity of experiments on general music similarity. Thus, is it important to minimize the ambiguity of the evaluation process and provide context or a scenario to allow users to provide more informed answers instead of asking vague questions about which track is more or less similar to the reference track.

In the context of music recommendation, there are many approaches to solve the cold-start problem [1], for exam-

Copyright: © 2022 Philip Tovstogan et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

ple, deep-learning and hybrid approaches [5, 6], or ones trying to predict the CF latent factors from audio [7, 8]. They all attempt to bridge the gap between CF and CB, thus requiring CF data to train the model. In this paper, we consider the scenario without personalization (anonymous user), i.e. where the system has no information about the user and needs to consider only track-to-track similarity.

Among the visualization interfaces of music collections, there are several commonly used techniques to reduce the dimensionality of the original latent spaces [2]. One of the first successful techniques is self-organizing maps (SOM) [9] used in Islands of Music [10] and other works that have followed and were inspired by it. In the more recent works, the newer algorithms such as t-SNE [11] and UMAP [12] gained popularity. They transform space in a non-linear way attempting to capture the relations between individual elements. The classic non-stochastic principal component analysis (PCA) approach can also be used [13]. While it is not as good at capturing the individual relationships between items, it captures the global structure of the whole space.

3. SIMILARITY METRIC

We aim to compare multiple latent spaces that contain the same set of items (music tracks). If we use one track as a reference and retrieve the nearest neighbors to the reference, we would have several different lists of nearest neighbors for each latent space. We introduce a simple metric S_n to calculate the similarity between two ranked lists of nearest neighbors L at the cutoff of n tracks that are obtained from two music similarity spaces X and Y. To differentiate this similarity between spaces from the music similarity that we also talk about, we use the term *NN-similarity* in this paper. We divide the number of tracks that are common in both lists by the cutoff to obtain the value between 0 (no common tracks) and 1 (all tracks are the same):

$$S_n(X, Y) = \frac{|L_{X,n} \cap L_{Y,n}|}{n} \tag{1}$$

If we consider the following example of $n = 5$ nearest neighbors to the track t_0 in the spaces X and Y, we would calculate the NN-similarity in the following way:

$$\begin{aligned} L_{X,5} &= (t_1, t_2, t_3, t_4, t_5) \\ L_{Y,5} &= (t_2, t_6, t_3, t_7, t_8) \\ L_{X,5} \cap L_{Y,5} &= \{t_2, t_3\} \\ S_5(X, Y) &= 2/5 = 0.4 \end{aligned}$$

S_n does not take into account the ranking: t_2 is ranked higher than t_3 in both $L_{X,5}$ and $L_{Y,5}$, but even if the relative rank would be reversed for $L_{Y,5}$, $S_5(X, Y)$ would still have the same value. In reality, if the cutoff n is much smaller than the number of tracks in the dataset ($n \in \{5, 10, 100, 200\}$), the primary difference between the lists is the number of intersected elements, not what is the difference between their ranks. The only potential benefit of using metrics that take ranks into account is to get the finer difference between lists that have the same amount of common tracks. We tried to use Spearman rank correlation or rank-based overlap (RDO) [14], and these metrics did not provide more information about the difference between pairs compared to simple S_n. [1]

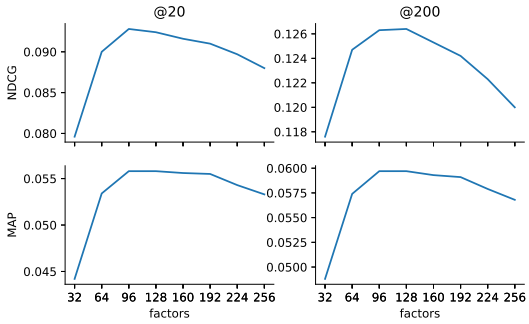


Figure 1. Baseline CF evaluation

4. DATA

Jamendo [2] is a platform that provides royalty-free music for commercial and personal use, including music streaming for venues or video production. In this paper, we use the tracks that are publicly available on their platform under Creative Commons (CC) licenses. In contrast to other datasets that provide audio available under CC licenses (FMA [15]), Jamendo ensures basic technical quality assessment. It was used before in the creation of open music datasets (MTG-Jamendo [16]).

4.1 Collaborative Filtering Features

The collaborative filtering data was provided as part of the collaboration with Jamendo and included 2.2 million interaction events (including plays, skips, etc.) that have associated numeric values assigned via an internal system for 170K tracks and 60K users. We pre-process the data by filtering out the tracks and users that had too few interactions (less than 5) and the top outliers, what results in 31K tracks and 27K users.

We do a pre-analysis of the data to determine the number of factors to be used for the matrix factorization. We use alternating least squares (ALS) algorithm [17] which is one of the SOTA matrix factorization algorithms. [3] Using a stratified split with a test ratio of 0.2 we evaluate different numbers of factors in terms of the performance using normalized discounted cumulative gain (NDCG) and mean average precision (MAP). The results are shown in Figure 1 with 96 factors providing the highest overall performance. We also consider 64 and 128 factors to compare the consistency of several CF spaces.

[1] See additional materials at the companion website `philtgun.me/deep-neighbors` for reports on other metrics.

[2] `jamendo.com`

[3] Implementation from `github.com/benfred/implicit`

Dataset		Architecture	Layer	Dim
MSD	×	MusiCNN	Embeddings	200
			Taggrams	50
MTAT		VGG	Embeddings	256
			Taggrams	50
AudioSet		VGGish	Embeddings	128

Table 1. Dimensions of latent spaces

4.2 Content-based Features

To extract content-based features we use the Essentia library [18] and the following music auto-tagging models [19]:

- *MusiCNN* [20] is a convolutional neural network (CNN) with vertical and horizontal convolutional filter shapes motivated by the music domain. It contains 6 layers and 787 000 trainable parameters.
- *VGG* is an architecture from computer vision [21] based on a deep stack of 3×3 convolutional filters that had been adapted for audio [22]. It contains 5 layers and 605 000 trainable parameters.
- *VGGish* is the original implementation of VGG architecture [21] with the number of output units is set to 3087 [23]. The number of trainable parameters is 62 million.

The models provided were pre-trained on several datasets. MusiCNN and VGG have been trained on top 50 tags from Million Song Dataset (MSD) [24] and MagnaTagATune (MTAT) [25]. These datasets contain music and are focused on the music auto-tagging. VGGish has been trained on AudioSet [26] which is an audio event recognition dataset that also includes music. This allows us to compare different architectures that have been trained on the same dataset and the same architecture trained on different datasets.

For the MusiCNN and VGG architectures, we consider the latent spaces constructed by the output layer (taggrams) and penultimate layer (embeddings). VGGish model only provides embeddings. The number of dimensions for the layers is summarized in Table 1. Thus, in total, we extract 9 content-based (CB) feature vectors.

We attempted to process the 31K tracks that we obtained from CF data, but due to some tracks being no longer available or corrupted, this number decreased to 29K.

4.3 Final Dataset

The final large dataset contains 29 275 tracks with successfully extracted CB features. We repeated the matrix factorization on the collaborative filtering data containing only those tracks (29 275 tracks × 27 235 users, 793 963 non-zero values) with the number of factors of 64, 96, and 128 to obtain the CF features. We release this final dataset with 3 CF and 9 CB representations publicly.

We create a smaller subset of the final dataset that is obtained by intersection with MTG-Jamendo [16] test set of split-0 which resulted in 1 372 tracks, which is comparable to a small music collection. We present the experiments on this small dataset, as it visualizes the relative differences between spaces better. [4]

Cutoff	5	10	100	200
Dataset (MSD vs. MTAT)	.26	.26	.46	.56
Arch. (MusiCNN vs. VGG)	.35	.36	.57	.65
Layer (emb. vs. tag.)	.50	.51	.68	.74

Table 2. Average NN-similarity along the variable

5. OFFLINE EXPERIMENTS

5.1 Latent Spaces

We compare the collaborative filtering and content-based spaces that are introduced in Section 4 in terms of NN-similarity S_n that was introduced in Section 3. We present the results using cosine distance to calculate nearest neighbors in Figure 2. Euclidean [5] and cosine distances produce very similar results, except that NN-similarity between CF rankings using Euclidean distance is lower.

The first thing that stands out in Figure 2 is that the CF spaces are quite dissimilar in terms of NN-similarity from CB spaces, as all pairs of rankings that include CF and CB spaces have the lowest values. This indicates that the music similarity captured by CF and CB spaces is noticeably different. The NN-similarity values between CF spaces stays consistent and is among the highest observed overall at all cutoffs. However, there is enough difference between CF spaces (e.g. max S_5 is 0.66 which means that 2 out of top-5 tracks will be different) to make the number of CF factors an important design decision.

Related to CB embeddings, at smaller cutoffs there is much more variability in the nearest-neighbors lists. Therefore, the choice of the latent space leads to significantly different outcomes in the use-cases that rely on the small number of nearest neighbors. For example, S_{10} varies between 0.16 to 0.58 which means that 4 to 9 tracks will be different between any two CB spaces. At larger cutoffs (100, 200) the NN-similarity between CB spaces is higher (S_{200} ranges from 0.46 to 0.77 between CB spaces).

We can calculate what choice impacts the NN-similarity more: dataset, architecture, or layer. To analyze this, we can fix two out of three variables and calculate the average NN-similarity between the pairs that come from comparing the third variable. For example, to determine how much the choice of *dataset* contributes to NN-similarity, we average the S values of MSD vs. MTAT for MusiCNN embeddings, taggrams, VGG embeddings, and taggrams. As we calculate those for a cutoff value of 5, we get that the average NN-similarity for choice of the dataset is 0.26, which means that if we change the training dataset, roughly only $0.26 \times 5 \approx 1$ track will be the same in the list of 5 nearest neighbors. The values for all cutoff values are presented in Table 2. According to the computed average NN-similarity, latent spaces produced by models trained

[4] The results on the large dataset are available on the companion website.

[5] More figures available on the companion website.

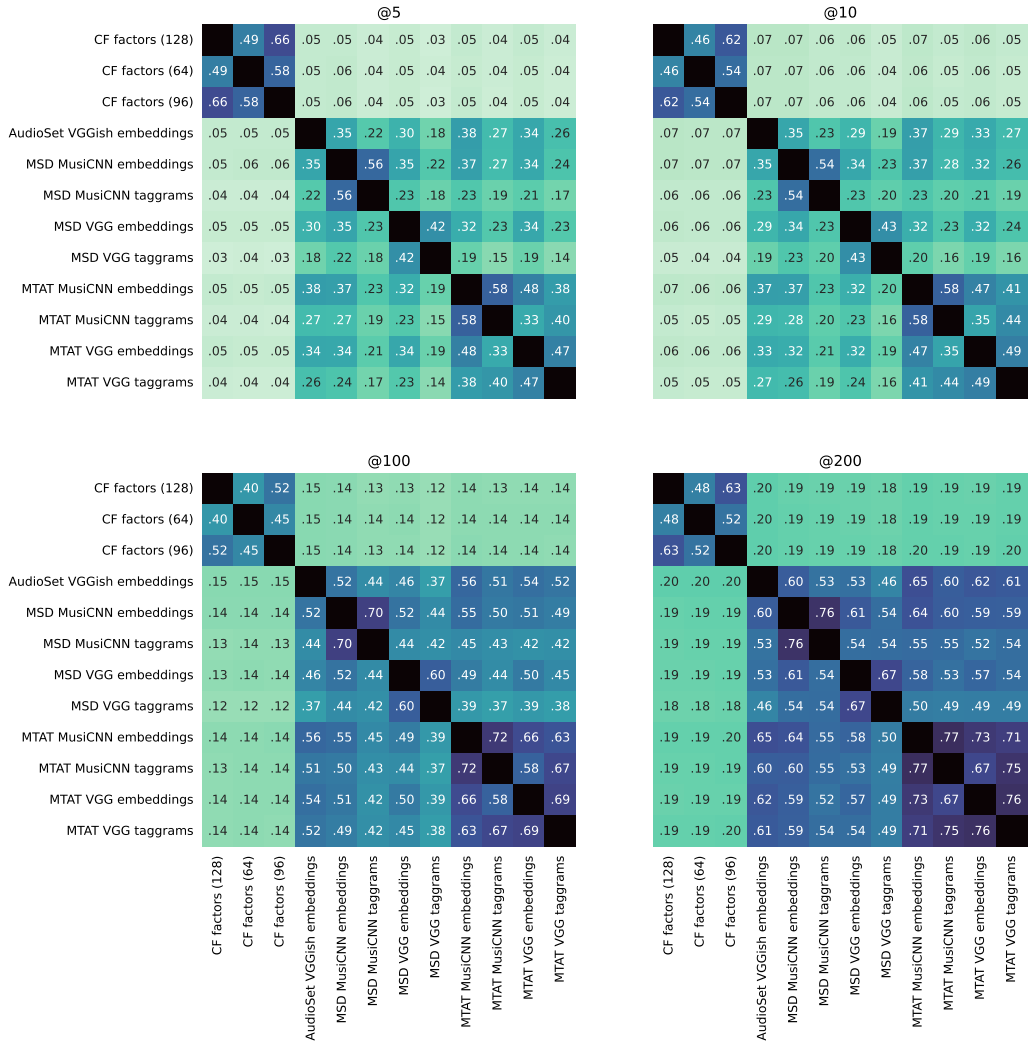


Figure 2. Nearest neighbor similarity (S_n) of CB vs. CF spaces

on different datasets (MSD vs. MTAT) are more dissimilar than the ones using different architectures (MusiCNN vs. VGG). Indeed MusiCNN and VGG are both CNN-based and share some similarities.

Regarding the choice of the layer (taggrams vs. embeddings), we can observe the highest NN-similarity when comparing spaces generated by the same model (same dataset and architecture). At the same time, taggram spaces are dissimilar to other CB spaces. That makes sense for spaces from different datasets, as the resulting tag spaces have different vocabulary and semantics. Interestingly enough, it also holds for different architectures on MSD (e.g. MSD MusiCNN vs. VGG taggrams $S_5 = 0.18$ which is close to MSD vs. MTAT MusiCNN taggrams: $S_5 = 0.19$). However, the NN-similarity is much higher for MTAT MusiCNN vs. VGG taggrams: $S_5 = 0.40$.

Overall the MTAT dataset seems to produce spaces that are in general more similar to each other compared to MSD. It can be attributed to the smaller size of MTAT, where the difference between architectures cannot be as pronounced. However, if the tag predictions produced by different architectures on the same dataset are close to each other, that might indicate the quality of annotations. While MSD annotations come from Last.fm folksonomy (every user can assign any tag), MTAT annotations come from the gamified system, where the annotators are encouraged to assign tags that might be similar to ones used by the other people [25].

Another interesting observation is that embeddings of different datasets and architectures, despite having higher dimensionality produce lists of nearest neighbors that are quite similar to each other (minimum values between CB embedding spaces: $S_5 = 0.30$, $S_{10} = 0.29$, $S_{100} = 0.46$, and $S_{200} = 0.53$). This is especially prominent at lower cutoff values (5, 10).

In the context of online evaluation with limited resources, it may be necessary to select a subset of latent spaces. Based on the results from Table 2, it makes sense to prioritize models trained on the different datasets rather than different architectures.

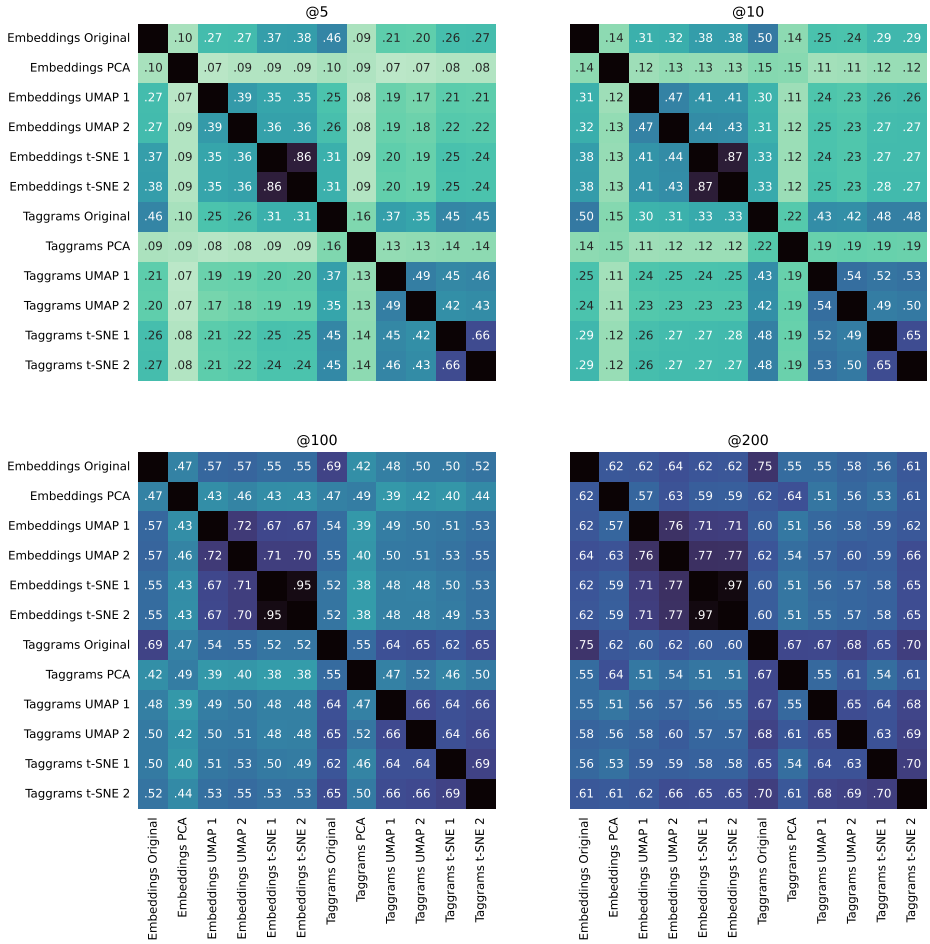


Figure 3. Nearest neighbor similarity (S_n) of different projections of MSD MusiCNN embeddings and taggrams

5.2 Projections

One of the applications of the latent spaces is to visualize the similarity between tracks. Hence, we want to use the same methodology to compare how well the NN-similarity is preserved while being projected on a 2D plane. Although some exploration interfaces use 3D planes, for consistency with previous research on music exploration [27] we only consider 2D. We consider PCA [28], t-SNE [11] and UMAP [12]. Moreover, as t-SNE and UMAP are stochastic, we consider two different seeds for each to measure the robustness. From previously obtained results it doesn't make sense to compare projections for different datasets or architectures, however, as embeddings and taggrams are quite similar to each other we consider both of them. Figure 3 shows the results for MSD MusiCNN embeddings and taggrams using Euclidean distance to calculate nearest neighbors, as it firstly, makes more sense to use in 2D, and secondly, cosine distance similarity is significantly lower for most pairs. [6]

Comparing different projections, it is evident from Figure 3 that t-SNE exhibits the highest NN-similarity to the original spaces. Nevertheless, this projection leads to noticeable changes in rankings (e.g. $S_{10} = 0.42$ for embeddings means that 6 tracks in the top-10 list will be different on average). UMAP is the second-best projection, with PCA being the poorest at preserving NN-similarity. The NN-similarity between different seeds for t-SNE is quite close to 1.0, which means that it is also more robust than UMAP. In general, at larger cutoff values the NN-similarity values are closer to each other. As PCA is a linear projection that works well in preserving the global structure of data without much consideration for nearest neighbors, its NN-similarity is quite low for small cutoff values (5, 10).

From a practical perspective, we see that using t-SNE for projection provides the best results and preserves more than 40% of nearest neighbors for small cutoffs. That means that in the visualization interface, among the 5 closest tracks in projected space 2 tracks are also closest in the original space to the reference track.

6. ONLINE EXPERIMENTS

Even if music similarity is quite subjective, it can be partially alleviated by asking more specific questions to the participants, as shown in related work in Section 2. We use a methodology similar to [29] to evaluate which spaces provide a better representation of music similarity for mu-

[6] Figures of cosine distance is available at companion website.

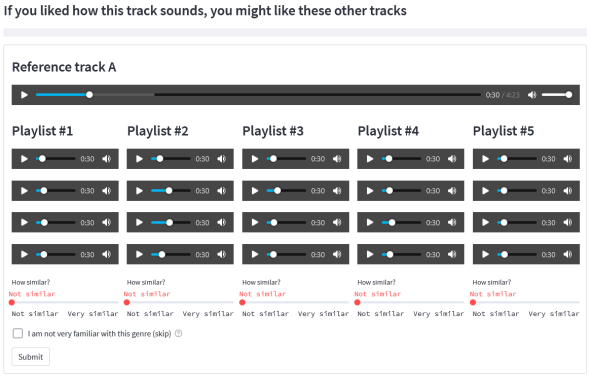


Figure 4. Online experiment interface

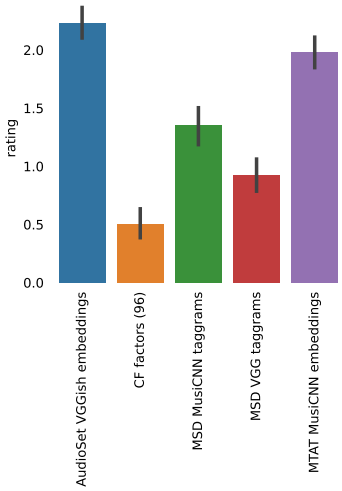


Figure 5. Online experiment results

sic recommendation. The difference with other studies is that this methodology evaluates the perception of playlists of top-N similar tracks, instead of individual comparisons of pairs of tracks. This approach is better aligned with tasks of music exploration and playlist generation.

We use the same small dataset for this experiment for consistency with offline experiments. The participants are presented with a reference track and several candidate playlists containing the nearest neighbors (ordered by their similarity to the reference) for each latent space. They are asked to rate the similarity of each playlist to the reference track in the hypothetical scenario of music recommendation: "If you liked how this track sounds, you might like these other tracks". The order of reference tracks is the same for all participants, while the playlists are presented in random order.

As the number of choices that can be presented to participants is limited by possible cognitive overload, we selected the 5 most dissimilar latent spaces: CF 96, MSD VGG taggrams, MSD MusiCNN taggrams, MTAT MusiCNN embeddings, and VGGish embeddings. We randomly select 4 reference tracks ensuring that they are quite different from each other and span several genres. In the interest of keeping the time to complete one instance of the experiment as low as possible while providing enough information to the participants, we decided to include 4 tracks in each playlist making it 21 tracks per reference track (including the latter) and 84 tracks in total. Because asking participants to listen to each track completely is unreasonable, by default we present the participant with a segment of 15 seconds that starts at 0:30 and ends at 0:45. However, the participants can use the controls of each player to play more different sections of the track if they feel that they need more information. Participants are encouraged to not spend much time on each track and use their intuition to rate the similarity, what is communicated explicitly in the instructions to the experiment. To measure the perceived similarity we provide a slider that uses a 4-point Likert scale: 0 - not similar, 1 - somewhat similar, 2 - quite similar, and 3 - very similar. We specifically avoided the neutral option to force participants to give their opinion. The interface of the experiment is shown in Figure 4.

We provide introductory text that explains the experiment, interface, purpose and allows the participants to continue with the experiment once they give their explicit consent. After circulating the link to the experiment [7] in the relevant communities (mailing lists, Twitter, subreddits [8]) we obtained data from 39 participants. We asked optional general demographic questions to verify coverage of different demographic groups. All participants are aged from 14–64 with the majority (53%) falling into the age group of 25–34. 55% of participants identify themselves as men, 33% as women, 9% non-binary and 3% preferred not to say. Concerning music background, 18% don't have any music training, 42% have some, 37% are hobbyists, and 3% (1) are professional musicians. The majority of participants (52%) listen to music on average 2–3 hours per day with the whole population listening from less than 1 hour up to 6–7 hours per day.

We use the Shapiro test (p-value$<$0.001) to verify the assumptions for the ANOVA test. We perform ANOVA (p-value$<$0.001) and Kruskal-Wallis (p-value$<$0.001) tests to verify if the choice of the latent space makes the similarity results significantly different. Subsequently, we use Tukey's honestly significantly differenced (HSD) test to identify which pairs of latent spaces are significantly different. The only pair of spaces where the difference is not significant is AudioSet VGGish vs. MTAT MusiCNN embeddings (p-value of 0.07).

Figure 5 shows the average ranking performance of each latent space with the standard deviation represented as a vertical line. We can see as both embedding spaces (AudioSet VGGish, MTAT MusiCNN) that we have chosen for the online experiment perform the best, with no statistical difference between them. An interesting observation is that AudioSet is a generic audio event recognition dataset, and the VGGish model trained on it performs comparably to the embeddings from the music auto-tagging dataset MTAT. MSD MusiCNN taggram space has a worse average similarity rating, with MSD VGG taggrams following

[7] `philtgun.me/similarity-experiment`
[8] `reddit.com/r/samplesize`

it. The poor performance of CF factors space can be attributed to the mismatch of the use-case, as it is intended to be used in conjunction with the user factors, not as a latent space. It is also possible that the small size of the dataset impacted the poor performance of CF factors.

The results show that content-based latent spaces can power the anonymous recommendation systems with a similarity that is rated at least as *quite similar*. This is a positive takeaway for exploration and visualization systems that can be built on top of similarity latent spaces.

7. CONCLUSIONS

We compared different collaborative filtering and content-based latent spaces in terms of the nearest-neighbor similarity. We observed that nearest neighbors obtained from CF spaces are very dissimilar to nearest neighbors obtained from CB approaches. Focusing on CB spaces, we identified that the choice of the training dataset (MSD vs. MTAT) tends to produce the most dissimilar spaces, followed by the architecture, and then layer. We observed that taggram spaces tend to be dissimilar across different datasets and architecture, while embedding spaces tend to be more similar. Interestingly, the consistency of CB latent spaces derived from a dataset may differ in terms of their nearest-neighbors similarity, as we observed on the example of the MTAT vs. MSD datasets. In the context of 2D visualization of latent spaces, t-SNE exhibits the highest nearest-neighbors similarity between original and projected spaces.

We performed an online experiment to evaluate a selection of dissimilar latent spaces in the context of music similarity for music recommendation. The results show that the CB spaces can be successfully used in music recommendation/exploration scenarios where user-generated data is absent due to design decisions. We observe that embedding spaces (AudioSet VGGish, MTAT MusiCNN) perform significantly better than taggram spaces (MSD MusiCNN, MSD VGG).

All analysis [9] and experiment interface [10] code is publicly available on GitHub, under Apache 2.0 license. The latent spaces are published on Zenodo [11] under CC BY-NC-SA 4.0 license, and the audio for the small dataset is available in MTG-Jamendo dataset. [12]

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068.

This work was conducted partially during the internship in Jamendo as part of the MIP-Frontiers project. We thank Jamendo for the collaboration, providing collaborative filtering data, and facilitation of the audio processing.

[9] github.com/philtgun/compare-embeddings
[10] github.com/philtgun/similarity-experiment
[11] zenodo.org/record/6010468
[12] mtg.github.io/mtg-jamendo-dataset

8. REFERENCES

[1] F. Ricci, L. Rokach, and B. Shapira, Eds., *Recommender Systems Handbook*, 3rd ed. Springer, Mar. 2022, in press.

[2] P. Knees, M. Schedl, and M. Goto, "Intelligent user interfaces for music discovery," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 165–179, Oct. 2020.

[3] A. Flexer, "On inter-rater agreement in audio music similarity," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan: Zenodo, Oct. 2014, pp. 245–250.

[4] A. Flexer, T. Lallai, and K. Rašl, "On evaluation of inter- and intra-rater agreement in music recommendation," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, p. 182, Nov. 2021.

[5] S. Oramas, O. Nieto, M. Sordo, and X. Serra, "A deep multimodal approach for cold-start music recommendation," in *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems (DLRS)*. Como, Italy: ACM, Aug. 2017, pp. 32–37.

[6] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proceedings of the 22nd ACM International Conference on Multimedia (MM)*. Orlando, FL, USA: ACM, Nov. 2014, pp. 627–636.

[7] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 26. Lake Tahoe, NV, United States: Curran Associates, Inc., Dec. 2013, pp. 2643–2651.

[8] A. Ferraro, Y. Kim, S. Lee, B. Kim, N. Jo, S. Lim, S. Lim, J. Jang, S. Kim, X. Serra, and D. Bogdanov, "Melon playlist dataset: A public dataset for audio-based playlist generation and music tagging," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Toronto, ON, Canada: IEEE, Jun. 2021, pp. 536–540.

[9] T. Kohonen, *Self-organizing maps*, 3rd ed., ser. Springer Series in Information Sciences. Berlin, Heidelberg: Springer, 2001, vol. 30.

[10] E. Pampalk, A. Rauber, and D. Merkl, "Content-based organization and visualization of music archives," in *Proceedings of the 10th ACM International Conference on Multimedia (MM)*. Juan-les-Pins, France: ACM, Dec. 2002, pp. 570–579.

[11] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, Nov. 2008.

[12] L. McInnes, J. Healy, and J. Melville, "UMAP: uniform manifold approximation and projection for dimension reduction," Feb. 2018.

[13] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg, "Embedding projector: Interactive visualization and interpretation of embeddings," Dec. 2016.

[14] W. Webber, A. Moffat, and J. Zobel, "A similarity measure for indefinite rankings," *ACM Transactions on Information Systems*, vol. 28, no. 4, pp. 1–38, Nov. 2010.

[15] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*. Suzhou, China: Zenodo, Oct. 2017, pp. 316–323.

[16] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, "The MTG-Jamendo dataset for automatic music tagging," in *Machine Learning for Music Discovery Workshop (ML4MD), International Conference on Machine Learning (ICML)*, Long Beach, CA, USA, Jun. 2019.

[17] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *2008 Eighth IEEE International Conference on Data Mining (ICDM)*. Pisa, Italy: IEEE, Dec. 2008, pp. 263–272.

[18] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra, "Essentia: An audio analysis library for music information retrieval," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*. Curitiba, Brazil: Zenodo, Nov. 2013, pp. 493–498.

[19] P. Alonso-Jiménez, D. Bogdanov, J. Pons, and X. Serra, "Tensorflow audio models in Essentia," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Barcelona, Spain: IEEE, May 2020, pp. 266–270.

[20] J. Pons and X. Serra, "MusiCNN: Pre-trained convolutional neural networks for music audio tagging," Sep. 2019.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations (ICLR)*. San Diego, CA, USA: arXiv, May 2015.

[22] K. Choi, G. Fazekas, and M. B. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*. New York City, NY, USA: Zenodo, Aug. 2016, pp. 805–811.

[23] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA, USA: IEEE, Mar. 2017, pp. 131–135.

[24] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, USA: Zenodo, Oct. 2011, pp. 591–596.

[25] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: the case of music tagging," in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*. Kobe, Japan: Zenodo, Oct. 2009, pp. 387–392.

[26] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA, USA: IEEE, Mar. 2017, pp. 776–780.

[27] P. Tovstogan, X. Serra, and D. Bogdanov, "Web interface for exploration of latent and tag spaces in music auto-tagging," in *Machine Learning for Music Discovery Workshop (ML4MD), International Conference on Machine Learning (ICML)*, Vienna, Austria, Jul. 2020.

[28] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, Nov. 1901.

[29] D. Bogdanov, J. Serrà, N. Wack, and P. Herrera, "From low-level to high-level: Comparative study of music similarity measures," in *2009 11th IEEE International Symposium on Multimedia (ISM)*. San Diego, CA, USA: IEEE, Dec. 2009, pp. 453–458.

INSIGHTS INTO TRANSFER LEARNING BETWEEN IMAGE AND AUDIO MUSIC TRANSCRIPTION

María Alfaro-Contreras **Jose J. Valero-Mas** **José M. Iñesta** **Jorge Calvo-Zaragoza**
Department of Software and Computing Systems, University of Alicante, Alicante, Spain
{malfaro, jjvalero, inesta, jcalvo}@dlsi.ua.es

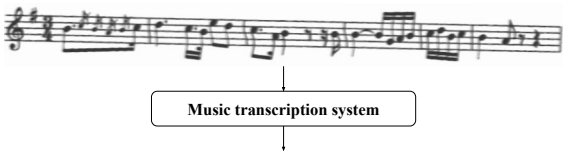
ABSTRACT

Optical Music Recognition (OMR) and Automatic Music Transcription (AMT) stand for the research fields that devise methods to transcribe music sources—documents or audio signals, respectively—into a structured digital format. Historically, they have followed different approaches to achieve the same goal. However, their recent definition in terms of sequence labeling tasks gathers them under a common formulation framework. Under this premise, one may wonder if there exist any synergies between the two fields that could be exploited to improve the individual recognition rates in their respective domains. In this work, we aim to further explore this question from a Transfer Learning (TL) point of view in the context of neural end-to-end recognition models. More precisely, we consider a music transcription system, trained on either image or audio data, and adapt its performance to the unseen domain during the training phase using different TL schemes. Results show that knowledge transfer slightly boosts model performance with sufficient available data, but it is not properly leveraged when the latter condition is not met. This opens up a new promising, yet challenging, research path towards building an effective bridge between two solutions of the same problem.

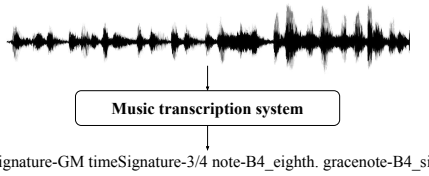
1. INTRODUCTION

The attainment of structured digital representations from music sources, typically known as *transcription*, stands as one of the major challenges in Music Information Retrieval (MIR) [1]. Within this community, there exist two main research lines that study how to computationally solve this problem when targeting either music documents—known as Optical Music Recognition (OMR) [2]—or acoustic music signals—namely, Automatic Music Transcription (AMT) [3]. Despite pursuing the same goal, these two fields have historically evolved in a disjoint manner since the differences in the nature of the data yielded specific task-oriented recognition frameworks, most commonly based on multi-stage processes [4].

However, some recent proposals in the MIR literature frame transcription problems under a sequence labeling formulation which approaches the task in a holistic or end-to-end manner [5]: the input data—either scores or acoustic pieces—are directly decoded into a sequence of music-notation symbols. Figure 1 graphically illustrates these music transcription approaches.



(a) Optical Music Recognition: music transcription using scores as inputs and a music-representation language as output.



(b) Automatic Music Transcription: music transcription using audio pieces as inputs and the same language as output.

Figure 1: End-to-end music transcription framework. OMR techniques deal with images and AMT techniques with audio signals; however, both tasks have to provide a result in a symbolic format that represents a score.

Currently, most end-to-end transcription approaches in the MIR literature resort to neural architectures [6, 7], which allows addressing OMR and AMT tasks with equivalent recognition models that only differ in the input data used for training the system. Furthermore, this common formulation enables exploring possible synergies that may exist between image and audio sources. Promoting such commonalities between both fields would open up a vast range of research avenues not previously explored in the related literature, such as: developing common language models, multimodal image and audio transcription, devising multi-task neural architectures capable of dealing with both tasks independently in a single model, or using pre-trained models with one modality and fine-tuning with the other. This latter case, which is commonly known as Transfer Learning, represents the focus of the work.

In a general sense, Transfer Learning (TL) addresses the case in which a model, initially trained with a particular source data distribution, is iteratively adapted to a differ-

Copyright: © 2022 María Alfaro-Contreras et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

ent, but related, target data domain [8]. Owing to a certain knowledge transference between the distributions, this process is expected to improve the recognition performance with respect to the case of exclusively training with the target domain [9]. While TL has been largely contemplated with successful results in other MIR tasks such as music classification [10] or vocal melody extraction [11], to our best knowledge no existing work has examined its use in transcription tasks due to the aforementioned limitations of legacy approaches.

This work studies the use of TL to exploit potential synergies between neural models of end-to-end image and audio transcription. More precisely, we examine the possible improvement achieved when training a recognition framework on a source domain—either image or audio—and transferring its knowledge to that not considered during the training stage. Such analysis may not only depict insights for improving the performance and robustness of the models, but it may also tackle the issue of data scarcity inherent to these fields by concurrently exploiting both domains [12, 13]. The results obtained show that, when a relatively large amount of training data is available, the proposed TL scheme achieves higher recognition rates than the case in which it is neglected. Moreover, this improvement is remarkably higher when adapting from AMT to OMR than the opposite case.

The rest of the paper is organized as follows: Section 2 develops the transfer learning framework considered; Section 3 describes the experimental setup; Section 4 presents and analyses the results; finally, Section 5 concludes the work and discusses possible ideas for future research.

2. METHODOLOGY

This section formally presents the neural end-to-end recognition and Transfer Learning frameworks contemplated in the work. To properly describe these design principles, we shall introduce some notation.

Let $\mathcal{T} = \{(x_m, \mathbf{z}_m) : x_m \in \mathcal{X},\ \mathbf{z}_m \in \mathcal{Z}\}_{m=1}^{|\mathcal{T}|}$ represent a set of data where sample x_m drawn from space $\mathcal{X}$ corresponds to symbol sequence $\mathbf{z}_m = (z_{m1}, z_{m2}, \ldots, z_{mN})$ from space $\mathcal{Z}$ considering the underlying function $g : \mathcal{X} \rightarrow \mathcal{Z}$. Note that the latter space is defined as $\mathcal{Z} = \Sigma^*$ where Σ represents the score-level symbol vocabulary.

Since we are dealing with two sources of information, we have different representation spaces $\mathcal{X}^i$ and $\mathcal{X}^a$ with vocabularies Σ^i and Σ^a related to the image scores and audio signals, respectively. In this regard, for the sake of clarity in the rest of the work, let $\mathcal{T}^i \subset \mathcal{X}^i \times \mathcal{Z}^i$ and $\mathcal{T}^a \subset \mathcal{X}^a \times \mathcal{Z}^a$ respectively represent the labeled sets of image scores and audio signals for training the transcription models.

2.1 Neural End-To-End Music Transcription

Due to its reported competitive performance in the related literature, we have considered a Convolutional Recurrent Neural Network (CRNN) scheme [14] with the Connectionist Temporal Classification (CTC) training algorithm [5] to approximate function g. This architecture comprises an initial block of *convolutional* layers devised to learn the adequate features for the particular recognition task followed by another group of *recurrent* stages which model the temporal/spatial dependencies of those features.

As commented, the network is trained using the CTC method as it allows training the CRNN scheme using unsegmented sequential data. In a practical sense, this mechanism only requires the different input signals to the scheme and their associated sequences of characters drawn from vocabulary Σ as its expected output, without any specific input-output alignment. It must be mentioned that CTC requires the inclusion of an additional "*blank*" symbol within the set of considered symbols, i.e., $\Sigma' = \Sigma \cup \{blank\}$ for enabling the detection of consecutive repeated elements.

Since CTC assumes that the architecture contains a fully-connected network of $|\Sigma'|$ neurons with a *softmax* activation, the actual output is a posteriogram with a number of frames given by the recurrent stage with $|\Sigma'|$ tokens each. Most commonly the final prediction is obtained out of this posteriogram using a *greedy* approach which retrieves the most probable symbol per step and a posterior squash function that merges consecutive repeated symbols and removes the *blank* label.

2.2 Transfer Learning Framework

As commented, Transfer Learning (TL) has been largely considered in the context of neural learning-based systems due to its reported benefits in terms of performance improvement. In a practical sense, such approaches typically initialize a recognition model using a source domain of data which is then fine-tuned with the actual target corpus, attending to a particular adaptation policy.

In this work, we explore the use of TL in neural end-to-end music transcription involving image scores and audio recordings given that, in both domains, we aim at retrieving a symbolic representation of the data at issue. More precisely, we assess the knowledge transference by posing these two scenarios: (i) pre-training a transcription model on one domain (either image or audio) and fine-tuning with the other one; and (ii) evaluating the amount of data necessary in the target domain for an effective transfer which outperforms the base case of neglecting TL. We shall now introduce some notation for then formally defining these scenarios.

Let CRNN^i and CRNN^a respectively denote two CRNN transcription models trained with a source domain of image, $\mathcal{T}^i$, and audio, $\mathcal{T}^a$, data, respectively. These initial models are adapted to a novel domain by being iteratively re-trained with the target data, hence obtaining CRNN$^{i \rightarrow a}$ for the image-to-audio scenario or CRNN$^{a \rightarrow i}$ for the audio-to-image one. Additionally, given that certain layers may be prevented from being updated during the domain transference stage—process typically known as *freezing*—we include subscript (L) to denote the range of L layers which are not affected by the re-training process, i.e., CRNN$^{i \rightarrow a}_{(L)}$ and CRNN$^{a \rightarrow i}_{(L)}$.

The first proposed scenario aims to uncover possible synergies between image and audio domains with a direct

knowledge transfer. In this regard, we assess the performance of the recognition models both when considering and disregarding pre-training. Two situations are posed attending to the target data: on the one hand, we compare model CRNN^{i} against CRNN$^{a\rightarrow i}_{(L)}$ for the OMR case; on the other hand, we confront CRNN^{a} against CRNN$^{i\rightarrow a}_{(L)}$ for the AMT situation. Moreover, for each of these cases we also assess the influence of freezing different layers when performing the TL process. Figure 2 graphically illustrates this scenario, particularly the case of training with image data, CRNN^{i}, and then performing TL to the audio domain while preventing the fifth layer of the model from being updated, i.e., CRNN$^{i\rightarrow a}_{(5:5)}$.

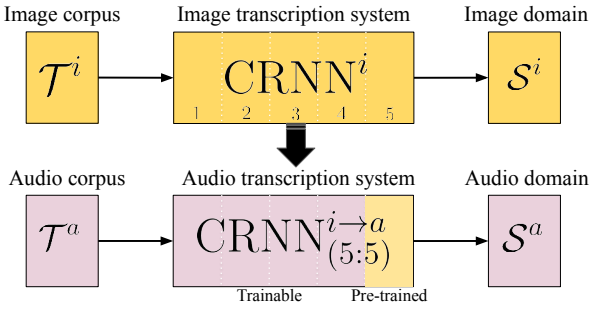


Figure 2: Graphical example of the Transfer Learning proposal: the CRNN^{i} image transcription model is adapted to the audio domain while preventing the parameters of the fifth layer from being updated: CRNN$^{i\rightarrow a}_{(5:5)}$.

It must be noted that, since the considered transcription models are based on deep neural networks, they generally require large amounts of labeled data to be trained [15]. This strong dependence on the size of the training set usually becomes an issue, especially when transcribing data domains with limited availability of transcriptions. Hence, as previously stated, TL poses itself as a promising alternative to solve the small-data problem by training the model on a certain domain for then transferring and adapting the knowledge acquired to the target one.

In this context, the second posed scenario simulates and studies this particular challenge attending to the target data domain: on one side, for the OMR case, CRNN^{i} is compared to CRNN$^{a\rightarrow i}_{(L)}$ when contemplating different subsets of the image corpus ($\mathcal{T}^{i-} \subset \mathcal{T}^{i}$); on the other side, when considering the AMT case, we compare CRNN^{a} against CRNN$^{i\rightarrow a}_{(L)}$ when considering subsets of the entire audio corpus ($\mathcal{T}^{a-} \subset \mathcal{T}^{a}$).

3. EXPERIMENTAL SETUP

This section presents the corpus considered, the definition of the different layers of the neural model, and the evaluation protocol used.

3.1 Corpus

We have considered the Camera-based Printed Images of Music Staves (Camera-PrIMuS) database [6]. This corpus contains 87,678 real music staves of monophonic incipits extracted from the *Répertoire International des Sources Musicales* (RISM). [1] For each incipit, different representations are provided: an image with the rendered score (both plain and with artificial distortions), several encoding formats for the symbol information, and a MIDI file.

In this regard, each transcription architecture considers a particular type of data: on the one hand, the OMR model takes as input the artificially distorted staff image of the incipit; on the other hand, for the AMT case, each MIDI file is synthesized with the FluidSynth software [2] and a piano timbre considering a sampling rate of 22,050 Hz, for then obtaining a time-frequency representation based on the Constant-Q Transform with a hop length of 512 samples, 120 bins, and 24 bins per octave, which is eventually embedded as an image that serves as the input. In both tasks, the height of the input figure considered is scaled to 256 pixels, maintaining the aspect ratio (thus, each sample might differ in width) and converted to grayscale, with no further preprocessing.

An initial data curation process was applied to the corpus to discard samples which may remarkably hinder the transcription, resulting in 67,000 incipits. [3] Since this reduced set still contains a considerably large amount of elements, we randomly selected a third of this curated set for our experiments, approximately, resulting in 22,285 incipits. Eventually, we derive three non-overlapping partitions—train, validation, and test—which correspond to the 60%, 20%, and 20% of the latter amount of data, respectively. Note that, since we are considering the same corpus for both image and audio data, both recognition tasks depict the same label space of $\Sigma^{i} = \Sigma^{a} = 1{,}166$ tokens.

3.2 Neural Network Configuration

While the presented sequence labeling paradigm allows considering a common formulation for both OMR and AMT tasks, in practice there is no universal neural architecture capable of achieving state-of-the-art performances in both cases. Generally, these configurations depend on a wide range of parameters which comprises the particular corpus considered, the amount of accessible data, or the available computational resources, among others.

Nevertheless, this work considers a common architecture for both transcription modalities for simplicity. In this regard, since neural end-to-end OMR models generally outperform AMT approaches [4], we consider a state-of-the-art architecture originally devised for the latter field for addressing both domains. Note that this is not a strong assumption since the performance decrease in the OMR domain with respect to the best achievable result is not remarkable and does not bias the conclusions obtained in the work. Hence, the actual composition of each layer is depicted in Table 1.

[1] Short sequence of notes, typically the first measures of the piece, used for indexing and identifying a melody or musical work.

[2] https://www.fluidsynth.org/

[3] This is the case of samples containing long multi-rests, which barely extend the length of the score image but take many frames in the audio signal.

Table 1: Layer-wise description of the CRNN model considered. Notation: Conv$(f, w \times h)$ stands for a convolution layer of f filters of size $w \times h$ pixels, BatchNorm performs the normalization of the batch, LeakyReLU(α) represents a Leaky Rectified Linear Unit activation with a negative slope of value α, MaxPool$(w \times h, a \times b)$ stands for the max-pooling operator of dimensions $w \times h$ pixels with $a \times b$ striding factor, BLSTM(n, d) denotes a bidirectional Long Short-Term Memory unit with n neurons and d dropout value parameters, Dense(n) means a fully-connected layer of n neurons, and Softmax$(\cdot)$ represents the softmax activation. Σ' denotes the considered alphabet, including the CTC-blank symbol.

Convolutional Block		**Recurrent Block**		**Classification Block**
Layer 1	**Layer 2**	**Layer 3**	**Layer 4**	**Layer 5**
Conv(8, 2×10)	Conv(8, 5×8)			
BatchNorm	BatchNorm	BLSTM(256)	BLSTM(256)	Dense($\lvert\Sigma'\rvert$)
LeakyReLU(0.20)	LeakyReLU(0.20)	Dropout(0.50)	Dropout(0.50)	Softmax$(\cdot)$
MaxPool(2×2)	MaxPool(1×2)			

All models in the work are trained using the backpropagation method provided by CTC for 100 epochs using the ADAM optimizer [16] with a fixed learning rate of 0.001 and a batch size of 4 elements.

3.3 Evaluation Protocol

We consider the Symbol Error Rate (SER) for assessing the performance of the presented recognition schemes as in previous works addressing end-to-end transcription tasks [6,7]. This figure of merit is computed as the average number of elementary editing operations (insertions, deletions, or substitutions) necessary to match the sequence predicted by the model with that in the ground truth, normalized by the length of the latter. Mathematically, this is expressed as:

$$\text{SER } (\%) = \frac{\sum_{m=1}^{|\mathcal{S}|} \text{ED}\left(\hat{\mathbf{z}}_m,\ \mathbf{z}_m\right)}{\sum_{m=1}^{|\mathcal{S}|} |\mathbf{z}_m|} \tag{1}$$

where $\mathcal{S} \subset \mathcal{X} \times \mathcal{Z}$ is a set of test data—from either the image or the audio domains—, ED : $\mathcal{Z} \times \mathcal{Z} \rightarrow \mathbb{N}_0$ represents the string Edit Distance [17], and $\hat{\mathbf{z}}_m$ and $\mathbf{z}_m$ denote the estimated and target sequences, respectively.

4. RESULTS

This section presents the results obtained for the different TL scenarios posed with the experimental scheme considered. For the sake of clarity, we analyze each TL case in a different section: a first one, denoted as *Scenario A*, that assesses the performance of the recognition models when TL is both considered and ignored, and a second one, namely *Scenario B*, that studies the amount of data required in the target domain for an efficient transfer process that outperforms the base case of ignoring TL. In all cases, the figures provided represent those obtained with the test partition when the validation data achieved its best performance. [4]

4.1 Scenario A: Fine Tuning

The first scenario posed studies the impact of TL on the recognition performance of the transcription models. For that, we consider base OMR and AMT recognition models—CRNN^i and CRNN^a, respectively—and perform a TL process for adapting them to their respective opposite domain, i.e., CRNN$^{i \rightarrow a}$ and CRNN$^{a \rightarrow i}$. Since this adaptation process may prevent different layers from being updated, we also analyze the influence of this parameter in the success of the task.

Table 2 reports transcription results in terms of the Symbol Error Rate (SER) when both base CRNN^i and CRNN^a models as well as CRNN$^{i \rightarrow a}$ and CRNN$^{a \rightarrow i}$ architectures are used for transcribing an evaluation set of image and audio data, respectively denoted as $\mathcal{S}^i$ and $\mathcal{S}^a$. Note that, as aforementioned, subscript (L) in the TL-based schemes indicates the range of layers in the recognition model (cf. Table 1) that are unaffected by the re-training process, being N/A the case in which all parameters are updated.

Inspecting the reported results, a first point to remark is the improvement achieved when training a recognition framework on a source domain—either image or audio—and transferring its knowledge to that not considered during the training stage when following the best re-training policy. On the image domain, CRNN$^{a \rightarrow i}_{(N/A)}$ reduces the error rate of CRNN^i over approximately 44% (from 9.58% to 5.31%); whereas, on the audio domain, CRNN$^{i \rightarrow a}_{(5:5)}$ obtains around a 3% of relative error improvement with respect to its baseline CRNN^a (from 29.24% to 28.32%). Such results support the initial hypothesis that TL may serve itself as a feasible strategy for obtaining more reliable and robust models.

The chosen re-training strategy plays an important role in the model performance. In this sense, when initially trained for addressing OMR tasks and adapted to AMT scenarios (CRNN$^{i \rightarrow a}$), the classification block may remain unaltered (CRNN$^{i \rightarrow a}_{(5:5)}$). However, learning first to solve the music transcription task with audio data (AMT) accounts for no model re-usability when transferring the knowledge to the image domain (OMR). Nonetheless, results indicate that knowledge transfer is better leveraged when adapting from AMT to OMR—CRNN$^{a \rightarrow i}$—than from OMR to AMT—CRNN$^{i \rightarrow a}$—since, on average, the former achieves better performance rates than the latter.

The last point to remark is that, as expected, when models are evaluated on a domain different than the one they were

[4] The code developed in the work is publicly available for reproducible research at: https://github.com/mariaalfaroc/smc-2022.git

Table 2: Results obtained in terms of the Symbol Error Rate (SER) for each transcription model considered when confronted to the image and audio domains, denoted as $\mathcal{S}^i$ and $\mathcal{S}^a$, respectively. Each column in the TL models—CRNN$^{i\rightarrow a}_{(L)}$ and CRNN$^{a\rightarrow i}_{(L)}$—denotes the range of L = [1:5] layers in the recognition model (cf. Table 1) which are not affected by the re-training process, being N/A the case in which all parameters are updated. Non-adaptive schemes—CRNN^i and CRNN^a—are provided for reference purposes. Best results for each evaluation domain are highlighted in bold type.

	CRNN^i	CRNN^a	Transfer learning: CRNN$^{i\rightarrow a}_{(L)}$					Transfer learning: CRNN$^{a\rightarrow i}_{(L)}$				
			N/A	5 : 5	4 : 5	3 : 5	2 : 5	N/A	5 : 5	4 : 5	3 : 5	2 : 5
$\mathcal{S}^i$	9.6	331.2	392.4	198.2	104.0	87.2	97.3	**5.3**	17.6	41.1	96.0	96.0
$\mathcal{S}^a$	98.9	29.2	28.6	**28.3**	40.3	86.7	93.9	98.4	97.6	95.6	96.0	96.0

trained on, their performance suffers a drastic deterioration. This downgrade is even more accused for audio models evaluated on image data. For TL-based approaches, this issue is referred to as catastrophic forgetting [18], as it implies the loss of previously learned information. We believe the multi-task learning paradigm could mitigate this problem by devising single architectures capable of solving several tasks, OMR and AMT, in this case.

4.2 Scenario B: Influence of the Target Set Size

As stated in Section 2.2, TL stands as a suitable solution for tackling the data scarcity in learning-based systems. This framework guarantees the convergence of the model on a sufficiently large data domain whose performance may be then adapted to a target one. We now explore this premise in the context of neural end-to-end music transcription systems.

To simulate this scenario, we consider different subset sizes of the target corpus. These divisions are used for adapting the base OMR and AMT recognition models trained on the entire source corpus—namely, CRNN^i and CRNN^a, respectively—to the target domain, hence respectively obtaining CRNN$^{i\rightarrow a}(5:5)$ and CRNN$^{a\rightarrow i}(N/A)$. Note that, for this evaluation scheme, only the best transfer configurations, obtained in Scenario A, are considered.

Table 3 reports the transcription results obtained for both the base and TL-oriented recognition models when evaluating on the same domain as the target space considered during the training stage. Note that train subset sizes reported range within $[50, 1000]$ since figures obtained above these values did not report any difference in performance.

Focusing on the image recognition task ($\mathcal{S}^i$), the use of TL denotes a better initialization of the model parameters as the CRNN$^{a\rightarrow i}_{(N/A)}$ model overcomes the base CRNN^i case when considering train sizes up to 100 samples. However, for a larger amount of training samples, the results report that the TL-based scheme severely degrades with respect to the base case. This impedes the use of previously acquired information in situations where it is most needed, such as those with a limited quantity of data. Nonetheless, since Table 2 shows that the inherent bias when training with audio data ($\mathcal{T}^a$) is overcome by fine-tuning with the entire image corpus ($\mathcal{T}^i$), we may assume the existence of a certain threshold in terms of train data elements in which the achievable error rate is lower than that when TL is disregarded (CRNN^i case).

In the case of audio transcription, TL leads to an improvement, of over 30%, only if we have a relatively small train set of 50 samples. When the training size is increased, the differences between considering or disregarding TL, are marginal, as observed in Tables 2 and 3. This suggests new TL methods should be devised to enhance the knowledge transfer when adapting from OMR to AMT.

5. CONCLUSIONS

The transcription of music sources into a structured digital format is one of the main challenges of the Music Information Retrieval (MIR) field. To tackle this problem, two research lines are considered: Optical Music Recognition (OMR), when considering visual input data such as scores, and Automatic Music Transcription (AMT), when considering acoustic input data such as audio signals. While these fields have historically evolved separately, their recent definition within a sequence labeling formulation results in a common representation for their expected outputs. This enables OMR and AMT tasks to be addressed with equivalent recognition models that only differ in their input modality.

Under this context, the following question naturally arises: is there any relationship between the learning features that each data-specific model acquires for solving the music transcription task? This paper provides insights into the posed topic by considering both modalities under a Transfer Learning (TL) scenario, which means adapting an already trained model for a new task. Specifically, we start from image transcription models and adapt them to audio transcription, and vice versa. To uncover the synergies and commonalities that can be established between the image and audio modalities, we analyze: (i) whether such knowledge transfer influences the overall recognition performance of the model; and (ii) how scenarios depicting data scarcity leverage this kind of relationship.

The obtained results reveal two relevant conclusions: on one end, TL boosts the model performance, being this improvement more pronounced when adapting from AMT to OMR; on the other end, the use of TL is not enough to overcome the scarcity of data as the aforementioned enhancement of the performance is only observable with sufficient target training data.

In light of these results, this work opens up new promis-

Table 3: Transcription results obtained in terms of the Symbol Error Rate (SER) for different subsets $\mathcal{T}^-$ of the train partition when considering domain-equivalent target and evaluation sets. The reported TL-based models—$\text{CRNN}^{a\to i}_{(N/A)}$ and $\text{CRNN}^{i\to a}_{(5:5)}$—for the image and audio target domains represent the best performing ones from the previous scenario whereas the non-adaptive schemes—CRNN^{i} and CRNN^{a}—are provided for reference purposes. Best performing figures for each evaluation domain and train corpus size are highlighted in bold type.

	Size of train corpus $\mathcal{T}^-$					
	50	75	100	250	500	1000
Evaluating on $\mathcal{S}^i$						
CRNN^{i}	95.3	95.2	94.9	**63.6**	**37.0**	**23.7**
$\text{CRNN}^{a\to i}_{(N/A)}$	**92.1**	**89.7**	**90.8**	79.5	77.6	49.9
Evaluating on $\mathcal{S}^a$						
CRNN^{a}	93.5	**56.7**	57.0	**45.9**	**40.4**	**37.5**
$\text{CRNN}^{i\to a}_{(5:5)}$	**64.0**	57.7	**53.6**	47.3	41.9	38.4

ing, yet challenging, avenues for research. For instance, the TL approach may be further explored by considering domain adaptation techniques. Besides, few-shot learning could be considered for tackling the data scarcity issue in this transcription paradigm. Finally, multi-task techniques may be explored to strengthen the synergies between image and audio music transcription by devising neural architectures capable of tackling both data domains in a single model.

Acknowledgments

This paper is part of the project I+D+i PID2020-118447RA-I00 (MultiScore), funded by MCIN/AEI/10.13039/501100011033. The first and second authors are supported by grants FPU19/04957 from the Spanish Ministerio de Universidades and APOSTD/2020/256 from "Programa I+D+i de la Generalitat Valenciana", respectively.

6. REFERENCES

[1] X. Serra, M. Magas, E. Benetos, M. Chudy, S. Dixon, A. Flexer, E. Gómez Gutiérrez, F. Gouyon, P. Herrera, S. Jordà *et al.*, *Roadmap for music information research*. The MIReS Consortium, 2013.

[2] J. Calvo-Zaragoza, J. Hajič Jr., and A. Pacha, "Understanding optical music recognition," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–35, 2020.

[3] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2018.

[4] C. de la Fuente, J. J. Valero-Mas, F. J. Castellanos, and J. Calvo-Zaragoza, "Multimodal image and audio music transcription," *International Journal of Multimedia Information Retrieval*, pp. 1–8, 2021.

[5] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 369–376.

[6] J. Calvo-Zaragoza and D. Rizo, "Camera-PrIMuS: Neural End-to-End Optical Music Recognition on Realistic Monophonic Scores," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, France, Sep. 2018, pp. 248–255.

[7] M. A. Román, A. Pertusa, and J. Calvo-Zaragoza, "A holistic approach to polyphonic music transcription with neural networks," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, Nov. 2019, pp. 731–737.

[8] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 3320–3328.

[9] Q. Yang, Y. Zhang, W. Dai, and S. J. Pan, *Transfer learning*. Cambridge University Press, 2020.

[10] L. Wang, H. Zhu, X. Zhang, S. Li, and W. Li, "Transfer learning for music classification and regression tasks using artist tags," in *Proceedings of the 7th Conference on Sound and Music Technology (CSMT)*. Springer, 2020, pp. 81–89.

[11] W.-T. Lu and L. Su, "Vocal melody extraction with semantic segmentation and audio-symbolic domain transfer learning," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, Sep. 2018, pp. 521–528.

[12] M. Alfaro-Contreras, D. Rizo, J. M. Iñesta, and J. Calvo-Zaragoza, "OMR-assisted transcription: a case study with early prints," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*. Online: ISMIR, Nov. 2021, pp. 35–41.

[13] L. Liu and E. Benetos, "From audio to music notation," in *Handbook of Artificial Intelligence for Music*. Springer, 2021, pp. 693–714.

[14] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, 2017.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[16] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[17] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet physics doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[18] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

SpecSinGAN: Sound Effect Variation Synthesis Using Single-Image GANs

Adrián Barahona-Ríos*
Department of Computer Science, University of York
Sony Interactive Entertainment Europe
ajbr501@york.ac.uk

Tom Collins
Department of Music, University of York
Music Artificial Intelligence Algorithms, Inc.
tomthecollins@gmail.com

ABSTRACT

Single-image generative adversarial networks learn from the internal distribution of a single training example to generate variations of it, removing the need of a large dataset. In this paper we introduce SpecSinGAN, an unconditional generative architecture that takes a single one-shot sound effect (e.g., a footstep; a character jump) and produces novel variations of it, as if they were different takes from the same recording session. We explore the use of multi-channel spectrograms to train the model on the various layers that comprise a single sound effect. A listening study comparing our model to real recordings and to digital signal processing procedural audio models in terms of sound plausibility and variation revealed that SpecSinGAN is more plausible and varied than the procedural audio models considered, when using multi-channel spectrograms. Sound examples can be found at the project website. [1]

1. INTRODUCTION

With the ever-growing complexity and length of video games and related media, balancing the quality and quantity of audio content has become a bigger challenge. As players may repeat actions in games, it is common that multiple sound files are used to sound design different audio variations of a single in-game interaction. Usually, this is done to prevent listener fatigue and avoid repetition [1], as well as to mimic reality (increase verisimilitude), where two sounds are hardly identical. To provide more variation to the sonic interactions, the sound design process regularly involves the use of different sound layers for a single sound effect [1]. For instance, a footstep sound effect can be broken down into the tip, the heel and the shoe fabric layers. Sound designers generally use audio assets from pre-recorded sound libraries and/or record the sounds on demand. Some sounds, however, may be rare, difficult or time-consuming to obtain, especially if multiple variations of them in the same style are needed. Considering some modern video games may have thousands of different sonic interactions within the game environment, the amount of assets required becomes an issue in the development process.

An alternative to using pre-recorded sound samples is the use of sound synthesis to source the sound assets. In the context of game audio, this is often called generative or procedural audio [2]. Procedural audio usually refers to the use of real-time digital signal processing (DSP) systems such as sound synthesisers, while generative (audio) can be defined as "algorithms to produce an output that is not explicitly defined" [3, p.1]. Apart from sound synthesisers, other DSP methods involve the manipulation of audio files in order to obtain a desired effect, transforming the source asset [4, 5]. Generative and procedural audio allow the dynamic creation of sound assets on demand. However, creating or using procedural or generative audio models may be challenging for sound designers. With the surge in interest in deep learning, novel sound synthesis techniques are being developed which can be seen as an alternative to traditional DSP methods such as [6], opening the possibility to synthesise sounds that may be challenging to achieve otherwise. Deep learning systems also blur the line between generative and procedural audio, with some architectures being able to synthesise audio on-demand with low latency [7] or even in real-time [8].

Here, we use single-image generative adversarial networks to synthesise novel variations of a single training sound effect. Generative adversarial networks (GANs) [9] are a generative modeling technique where – typically – two neural networks (generator and discriminator) compete against each other. The generator tries to capture the data distribution by fooling the discriminator, and the discriminator tries to distinguish between real training data and the output produced by the generator. The end goal consists of having a generator capable of producing plausible novel content with a similar distribution as the training data, and this usually involves training on large datasets over extended periods of time. On the other hand, single-image GANs, such as SinGAN [10], are trained on overlapping patches of a single training image, modeling its underlying internal distribution and producing novel samples with similar visual content, but without the need of a large dataset. We apply the same principle to the audio domain, resulting in an alternative way to generate sound assets.

2. RELATED WORK

There are multiple studies on the use of DSP systems for the synthesis of sound effects, such as in the synthesis of footsteps [11] or aeroacoustic sounds [12]. DSP-based

*This work was undertaken during an internship at Sony Interactive Entertainment Europe.

[1] www.adrianbarahonarios.com/specsingan

Copyright: © 2022 Adrián Barahona-Ríos et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

synthesis of sound effects has been demonstrated to be perceptually effective when suitable sound synthesis methods are used [13]. Guidelines to choose a suitable sound synthesis method to synthesise a target sound have been proposed also [2] [14] [15]. There have also been DSP systems catered to generate variations of target pre-recorded impact sounds, such as in [4] and, after we conducted our listening study, in [5].

Audio synthesis using deep learning, often called neural audio synthesis, is a very active field of research. These techniques usually work either directly on the time domain (waveforms) or in the frequency domain (spectrograms). There are multiple architectures and applications that have been explored, such as autoregressive models for waveform synthesis [16], GANs for unconditional [17] and conditional [18] waveform synthesis or for frequency-domain conditional synthesis [19] [20] or diffusion models for conditional and unconditional waveform synthesis [21]. Other architectures, such as Differentiable DSP (DDSP) [22], incorporate DSP methods like spectral modeling synthesis [23] into the deep learning domain. All these architectures need a training dataset of some sort.

Single-image GANs exploit the internal statistics of a single training example to generate novel variations from it. An example of these architectures is SinGAN [10]. SinGAN is an unconditional generative model that uses a progressive growing multi-scale approach, training a fully convolutional GAN on a different resolution at each stage. The model starts producing small-sized images, which are upsampled and fed to the next stage alongside a random noise map. SinGAN uses patch-GANs [24], training on overlapping patches of the training image at the different stages. Given the GAN receptive field is fixed with respect of the image size, the model learns to capture finer details as the training progresses. This fully convolutional design also enables image generation of arbitrary size just by changing the dimensions of the input noise maps. ConSinGAN [25] is another single-image GAN architecture. Built upon SinGAN, the authors proposed some improvements to it, such as concurrent training of the different stages or the resizing approach when building the image "pyramid" for the different resolutions, reducing the number of parameters and the training time. These architectures are also capable of performing other tasks such as retargeting, animation or super-resolution. There are several other single-image architectures, such as Hierarchical Patch VAE-GAN [26], which is capable of producing not only images but videos. Other single-image methods, such as Drop the GAN [27] use patch-nearest-neighbors instead.

Catch-A-Waveform (CAW) [28] is a recent audio time-domain architecture inspired by single-image GANs that is capable of producing novel audio samples of arbitrary length with just 20 seconds of training data. They showcase the architecture's performance on music, speech and environmental sounds (such as applause or thunderstorm), yielding promising results. CAW is also capable of performing different tasks directly on the audio domain, such as bandwidth extension, denoising or audio inpainting. In our case, we focus the modeling at the individual sound effect level with the aim of producing novel one-shots instead of streams of audio such as music excerpts or speech.

3. METHOD

3.1 Audio Representation

While a 2-channel frequency-domain representation consisting of a magnitude spectrogram and instantaneous frequency (IF) has been used to achieve state-of-the-art results on GAN audio synthesis of pitched musical notes [19] [29], recently, [30] studied the use of 1-channel phaseless log-magnitude spectrograms as an alternative for synthesising non-harmonic sounds (such as chirps or pops), achieving better perceptual results in this context. To invert the spectrogram back to audio, they reconstruct the phase using the Phase Gradient Heap Integration (PGHI) algorithm [31]. Another popular phase reconstruction method is the Griffin-Lim [32] algorithm. We tested a phaseless log-magnitude spectrogram representation with both the PGHI the Griffin-Lim algorithms and, in our preliminary tests, Griffin-Lim produced better perceptual results using a 75% frame overlap. The FFT size choice also has a significant impact on the results, being 512 the size that produced the best consistent results in our tests. We opted then to use a phaseless log-magnitude spectrograms with a FFT size of 512, 75% overlap, a Hanning window of the same size of the FFT and reconstructing the phase with Griffin-Lim.

Similar to [33], we use multi-channel spectrograms as the input to our model. While in [33] the authors use the multi-channel spectrograms to represent the pitch and intensity of musical notes, we use them to represent the different layers of the sound effect. To be precise, multi-channel spectrograms are built by stacking multiple sound layers along the channel axis, and these layers are provided by the user to the network directly (they are not extracted programmatically from layered sounds). We use the multi-channel spectrogram term (instead of multi-layer) for consistency with the literature. The use of multi-channel spectrograms instead of single-channel spectrograms in the context of sound effects synthesis presents some benefits. For instance, they allow for some parametrisation of the sound synthesis as the layers could be synced to an animation, triggered asynchronously, have different volume from each other, etc.

To build the multi-channel spectrograms first we load and normalise the sound layers in a range of $[-1, 1]$. Next, we measure the longest sound effect layer in the time-domain and zero-pad the remaining layers to this length. Then, we transform the multiple audio layers into log-magnitude spectrograms, discarding the phase information and stacking them along the tensor channel axis as if they were different channels of an image. Finally, we normalise the spectrogram (mean 0, standard deviation 1). During inference we revert this normalisation, invert the logarithm, permute the channel and batch dimensions (so the layers appear as different sounds in a batch) and transform the spectrograms back to audio. If the training sound effect has only one layer, a single-layer spectrogram is used. Unless stated otherwise, all sound layers are mono, with a sampling rate of $44.1\,\mathrm{kHz}$.

3.2 Architecture

SpecSinGAN is built upon the ConSinGAN [25] architecture. As depicted in Figure 1, the SpecSinGAN discriminator will have as many input blocks in parallel as the number of channels (sound effect layers) of the training spectrogram, each one of them consisting of a convolutional layer followed by a Leaky ReLU activacion. For instance, for a 3-channel spectrogram, the discriminator will have 3 input blocks, and each channel of the spectrogram will be processed individually by just one of them in parallel. The resulting feature maps are stacked along the batch axis, in a somewhat similar manner to [33]. Then, the feature maps go onto 3 groups of convolutional layers followed by Leaky ReLU activations, and finally onto a last convolutional layer. All the convolutional layers have a kernel size of 3, stride 1 and 1 dilation rate. We use a 0.05 alpha value for the Leaky ReLU non-linearity. As suggested by the ConSinGAN authors [25], we implemented a second discriminator for the final training stages, finding slight improvements on the results. The second discriminator is identical to the first one, but with dilation on all its convolutional layers to increase its receptive field.

The SpecSinGAN generator is similar to the growing generator in ConSinGAN. For the first training stage, the generator has 4 convolutional layers, each one of them followed by batch normalisation and a Leaky ReLU activation, and a final output convolutional layer. As we do not constrain the range of the spectrograms, we do not use a hyperbolic tangent activation in the output of the generator. Each training stage, 3 more convolutional layers with batch normalisation and Leaky ReLU activations are added just before the output convolutional layer, increasing the generator capacity as the training progresses. All layers have the same hyperparameters as the layers in the first discriminator.

By default, we set the training stages in the training process to 10. At the start, an 'image pyramid' of training spectrograms is built, going from the first stage where the images are downsampled to a coarser scale, to the last stage, where the images are at the original resolution. We set the maximum size of the image to their original resolution, and the minimum size depending on the sound. The training starts at the coarsest scale, where the generator receives the noise maps corresponding to that stage and produces spectrograms at that size, while the discriminator classifies their different overlapping patches against the spectrogram from the training pyramid at that scale. As can be seen in Figure 3.2, at the consecutive stages and until reaching the last one, the features maps from previous stages are upsampled and passed directly to the current stage, mixing them with the corresponding noise at that scale to increase diversity. We set the default noise amplification (a multiplier to weight the noise against the feature maps) to 0.1. The feature maps from the previous stage are also upsampled and summed after the convolutional block at each stage. Overall, the training process is identical to ConSinGAN, with the exception we add a second discriminator halfway through training.

As with [25], we train 3 stages concurrently, using a learning rate of 0.0005 for the current stage and 0.1 scaling for the other 2 stages below. We use WGAN-GP [34] for the adversarial loss, in combination with a reconstruction loss with a weight of 10. We also use the ConSinGAN upsamplig strategy for the generator, where the feature maps after the first convolutional block are slightly upsampled to increase the diversity at the edge of the images (refer to [25] for details). More details of the training hyperparameters can be found in Section 4.

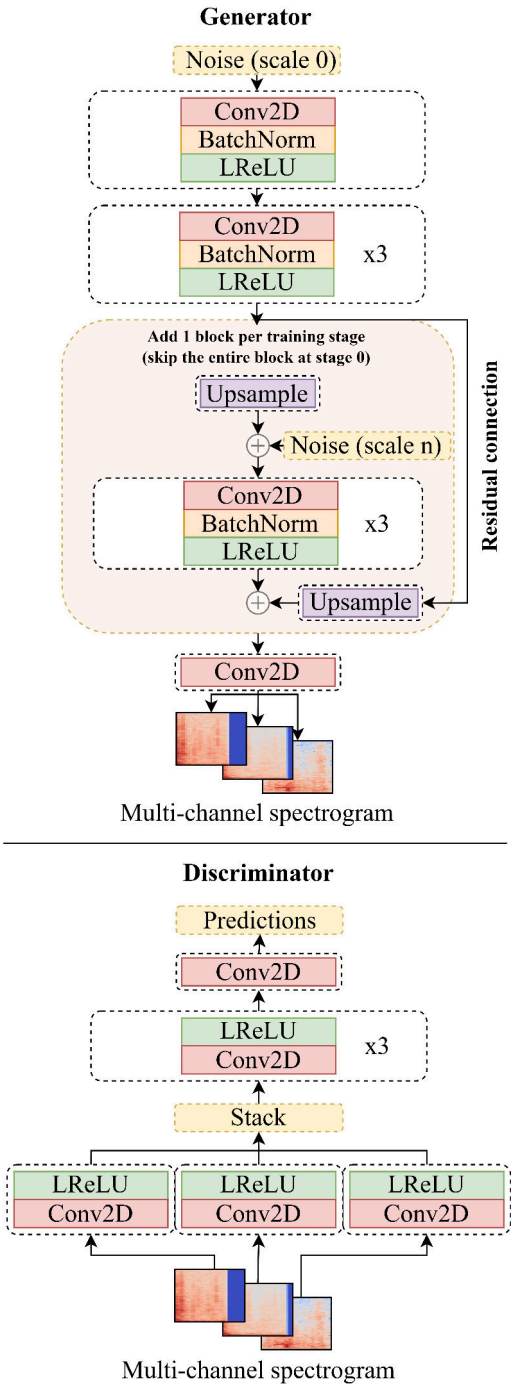


Figure 1. SpecSinGAN architecture. For simplicity, the internal upsampling and padding operations in the generator are not represented in the image (details in Section 3.2).

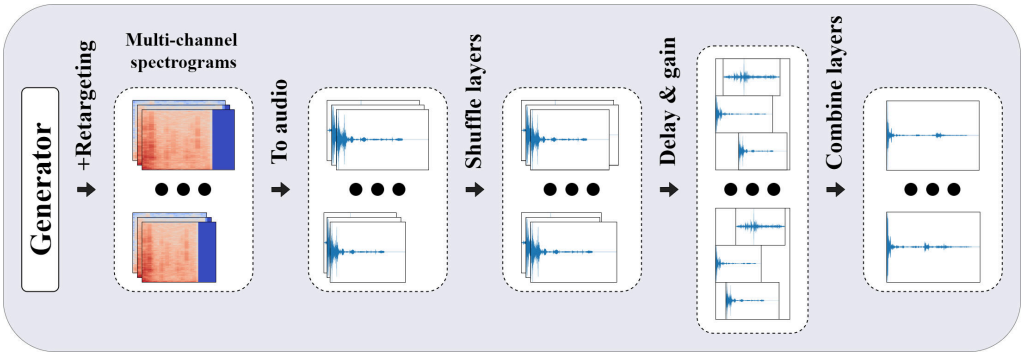


Figure 2. Multi-channel synthesis during inference: the generator produces multi-channel spectrograms that are slightly different in length on their x-axis. They are transformed back to audio using the Griffin-Lim algorithm [32], shuffling the different length layers afterwards. We finally apply a randomised delay and gain to the individual layers that comprise the sound effect, combining them to render the final audio files.

As depicted in Figure 2, during inference we randomise the x-axis dimension of the noise maps that go into the generator. Note we do not randomise the x-axis dimension of the noise maps during training, but only during inference once the model is trained. As a result, each multi-channel spectrogram of the batch has a different length. This is thanks to the fully-convolutional nature of the network, where the input noise maps are not constrained to be of a specific shape. We transform the spectrograms into audio and shuffle the layers of the batch, combining the different-length layers. Finally, we randomise the delay and gain of the layers with respect to each other and sum the layers into a single audio file.

4. EXPERIMENTS

We selected four one-shot sound effect categories that are commonly found in video games and media, using three sound layers for each of them: footsteps on concrete (heel, tip and shoe fabric), footsteps on metal (heel, tip and metal rattle), gunshots (noise/body, mechanic component and tail) and character jump (human efforts, foley of the character clothes and metal clinks of character equipment). We collected the training sound effects from the Freesound website [35]. To assess the effect of the multi-channel spectrogram approach, we trained two models per sound effect: one with single-channel spectrograms and another with multi-channel spectrograms. The models trained with single-channel spectrograms use a single training audio file with all the different layers combined.

SpecSinGAN allows the input of both arbitrary length audio files and arbitrary numbers of layers. However, we found that different sound effects require different training hyperparameters, depending on the number of layers, the shape of the training sound, the frequency content, and the desired degree of variation. For our purposes, three layers of relatively short one-shots (≈200-750 ms) are a good compromise. In general, more layers or reverberant sounds will require a higher number of iterations per training stage. For sounds with very sharp transients at the beginning, adding a small zero-padding at the start of the audio file before training can prevent artifacts. Other parameters could be changed to accommodate longer or more challenging sounds, such as the the number of training stages or even the sampling rate or the FFT size.

In our experiments, we trained both the single and multi-channel spectrograms of the footsteps on concrete and metal for 2,000 iterations per stage, using 64 filters in the convolutional layers, a dilation rate of 3 in the second discriminator and setting the minimum tensor size of the training pyramid (on any axis) to 50. Gunshots were trained for 8,000 iterations per stage using 128 filters, a dilation rate of 2 in the second discriminator and setting the minimum size to 11. Finally, the character jumps were trained for 8,000 iterations per stage using 128 filters, a dilation rate of 3 in the second discriminator and setting the minimum size to 25. As an indication, training on an NVIDIA Tesla V100 took approximately 50 minutes for single and multi-layer footsteps on concrete and metal, 200 minutes for single-layer gunshots, 600 minutes for multi-layer gunshots and 240 and 500 minutes for single and multi-layer character jumps respectively.

During inference we randomise the delay and gain of the different layers so they are coherent with the aesthetics of the final sound effect. In terms of retargeting, we apply no more than a randomised $\pm 15\%$ range multiplier to the x-axis (corresponding with the spectrogram time axis) of the input noise maps. Multiplying the input noise maps by a larger number may result in audible artifacts. For reference, synthesising 1,000 sounds on a NVIDIA Tesla V100 took approximately 60 and 100 seconds for single and multi-channel footsteps in concrete respectively.

5. EVALUATION & RESULTS

We designed a listening study to evaluate the sounds resulting from the experiments in Section 4, for the 4 catgeories of footsteps on concrete, metal, gunshot, and character jump. We compared 4 different systems on plausibility and variation: real recordings (hereafter, Real), SpecSinGAN with single and multi-channel spectrograms (denoted by the subscript "s" or "m" respectively), and Nemisindo [6, here-

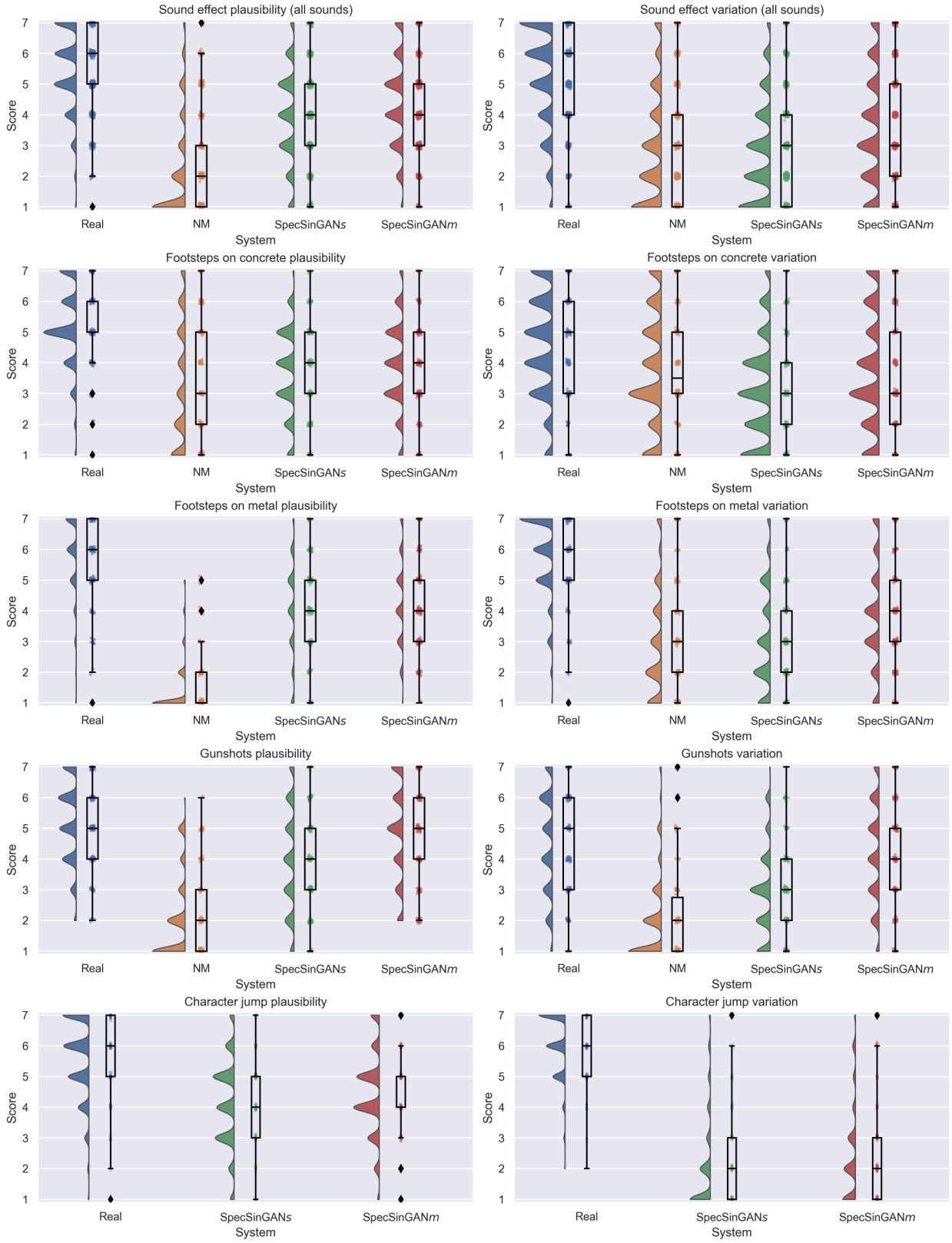


Figure 3. Listening study results for the different sound effects and systems considered. Real recordings are denoted by 'Real', Nemisindo by 'NM' and SpecSinGAN using either single or multi-channel spectrograms by 'SpecSinGAN$_s$' and 'SpecSinGAN$_m$' respectively. Participants were asked to rate each stimuli on both plausibility and variation in a $[1..7]$ scale. Note the scatter plot represents the individual ratings, with jitter added to prevent overlapping as only natural numbers were given as rating options.

after, NM]. NM is a web-based procedural audio service that enables the creation of synthesised sound effects using DSP methods. For the character jump sounds, we only compared Real and SpecSinGAN variants, as NM does not offer this type of sound at the time the study was carried out. Sounds for the Real category were sourced from the same Freesound [35] training examples used in Section 4, taking sound variations from the same file or designing them (e.g., cutting, equalising, fading) when needed. The SpecSinGAN sounds are those of Section 4. NM sounds were taken directly from their presets (without searching the parameter space), finding the closest ones to the target category of sounds.

The sounds were presented concatenated to make sound actions (e.g., footsteps turned into walking), resulting in audio clips approximately 5-seconds long. Participants rated 5 sounds per category per system in terms of plausibility and variation on a scale of 1 (not at all plausible/varied) to 7 (completely plausible/appropriately varied). We used the Prolific platform to conduct the listening test and recruited 30 participants, compensating them £9/h. We pre-screened participants such that only those over 18 years old who play video games for at least 6 hours a week were selected, and encouraged them to use headphones during the evaluation. The listening study results are shown in Figure 3.

We used non-parametric Bayes factor analysis (BFA) to investigate our hypotheses about how the systems would compare in a listening study. This is because the rating data cannot be assumed to be normally distributed, and because Bayesian hypothesis testing is widely regarded as superior to the frequentist variety, with the former allowing for finding evidence in favour of "no difference between systems" if the data suggest as much [36]. We hypothesised that Real would have better plausibility and variation than any other system, and the BFA found extreme evidence for this ($BF_{10} > 100$). We also hypothesised SpecSinGAN would have slightly more plausibility than NM for footsteps on concrete, finding anecdotal and strong evidence for this for SpecSinGAN$_{\text{s}}$ ($BF_{10} = 1.09$) and SpecSinGAN$_{\text{m}}$ ($BF_{10} = 12.5$) respectively. In addition, we hypothesised that SpecSinGAN would have higher plausibility ratings than NM for footsteps on metal, and the BFA found extreme evidence for this ($BF_{10} > 100$). Finally, we hypothesised SpecSinGAN and NM would have similar plausibility for gunshots, and the BFA rejected this ($BF_{10} > 100$). The data suggested SpeSinGAN having more plausibility than NM for gunshots. Regarding variation, we hypothesised SpecSinGAN and NM would have similar values, confirmed for SpecSinGAN$_{\text{s}}$ according to the BFA ($BF_{10} = 0.09$) and rejected for SpecSinGAN$_{\text{m}}$ ($BF_{10} > 100$). In the latter case, the data suggest SpecSinGAN$_{\text{m}}$ had higher variation values than NM.

6. DISCUSSION

Audio asset creation can be a time-consuming process. In this paper we presented SpecSinGAN, an unconditional generative architecture capable of synthesising novel variations of one-shot sound effects with a single training example.

SpecSinGAN performed statistically significantly better in the listening study, compared to the procedural audio models considered. We used the NM presets available at the time of the listening study instead of searching the parameter space, so, while this is a reasonable choice, it is possible that NM (or any other DSP tool) would produce better results following a more exhaustive search of its parameter space.

In future work, we plan to introduce user control over the synthesis, alongside increasing its plausibility and variation to be on par with real recordings. Another improvement would be the implementation of automatic hyperparameter tuning, given that, as discussed in Section 4, different sounds required different hyperparameters and, despite there being some intuitions on how to tune them, this still involves a manual process.

We would also observe that, despite showing SpecSinGAN is a viable alternative to synthesise arbitrary one-shot sound effects, DSP-based systems are also capable of producing continuous streams of audio, as well as running in real-time with direct input from either human-interpretable controls or in-game parameters, granting them great adaptability. We also acknowledge that, while we focused on arbitrary sound effects, further listening studies need to be carried out to understand how SpecSinGAN compares to DSP methods such as [5] for generating variations of target percussive sounds and to [28], adapting it to work with shorter sounds at $44.1\,\text{kHz}$. We suggest, however, that SpecSinGAN can be useful in a contexts where 1) sound designers need to produce novel variations of a specific pre-recorded sound, or 2) data is scarce, in which case SpecSinGAN acts as a data augmentation tool.

Acknowledgments

This work was supported by the EPSRC Centre for Doctoral Training in Intelligent Games & Game Intelligence (IGGI) [EP/L015846/1] and Sony Interactive Entertainment Europe.

7. REFERENCES

[1] G. Zdanowicz and S. Bambrick, *The Game Audio Strategy Guide: A Practical Course*. Focal Press, 2019.

[2] A. Farnell, *Designing Sound*, 2010.

[3] M. Yee-King and I. Dall'Avanzi, "Procedural Audio in Video Games," 2018.

[4] D. B. Lloyd, N. Raghuvanshi, and N. K. Govindaraju, "Sound Synthesis for Impact Sounds in Video Games," in *Symposium on Interactive 3D Graphics and Games*, 2011, pp. 55–62.

[5] J. Fagerström, S. J. Schlecht, V. Välimäki *et al.*, "One-to-Many Conversion for Percussive Samples," in *International Conference on Digital Audio Effects*, 2021, pp. 129–135.

[6] P. Bahadoran, A. Benito, T. Vassallo, and J. D. Reiss, "Fxive: A Web Platform for Procedural Sound Syn-

thesis," in *AES Convention 144*. Audio Engineering Society, 2018.

[7] M. Chang, Y. R. Kim, and G. J. Kim, "A Perceptual Evaluation of Generative Adversarial Network Real-Time Synthesized Drum Sounds in a Virtual Environment," in *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. IEEE, 2018, pp. 144–148.

[8] F. Ganis, E. F. Knudesn, S. V. Lyster, R. Otterbein, D. Südholt, and C. Erkut, "Real-time Timbre Transfer and Sound Synthesis using DDSP," *arXiv preprint arXiv:2103.07220*, 2021.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[10] T. R. Shaham, T. Dekel, and T. Michaeli, "SinGAN: Learning a Generative Model From a Single Natural Image," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4570–4580.

[11] R. Nordahl, S. Serafin, and L. Turchet, "Sound Synthesis and Evaluation of Interactive Footsteps for Virtual Reality Applications," in *2010 IEEE Virtual Reality Conference (VR)*. IEEE, 2010, pp. 147–153.

[12] R. Selfridge, D. Moffat, E. J. Avital, and J. D. Reiss, "Creating Real-Time Aeroacoustic Sound Effects Using Physically Informed Models," *Journal of the Audio Engineering Society*, 2018.

[13] D. Moffat and J. D. Reiss, "Perceptual Evaluation of Synthesized Sound Effects," *ACM Transactions on Applied Perception (TAP)*, vol. 15, no. 2, pp. 1–19, 2018.

[14] A. Misra and P. R. Cook, "Toward Synthesized Environments: A Survey of Analysis and Synthesis Methods for Sound Designers and Composers," in *ICMC*, 2009.

[15] P. R. Cook, *Real Sound Synthesis for Interactive Applications*. CRC Press, 2002.

[16] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A Generative Model for Raw Audio," *arXiv preprint arXiv:1609.03499*, 2016.

[17] C. Donahue, J. McAuley, and M. Puckette, "Adversarial Audio Synthesis," *arXiv preprint arXiv:1802.04208*, 2018.

[18] A. Barahona-Ríos and S. Pauletto, "Synthesising Knocking Sound Effects Using Conditional WaveGAN," in *17th Sound and Music Computing Conference, Online*, 2020.

[19] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial Neural Audio Synthesis," in *International Conference on Learning Representations*, 2018.

[20] J. Nistal, S. Lattner, and G. Richard, "DrumGAN: Synthesis of Drum Sounds With Timbral Feature Conditioning Using Generative Adversarial Networks," *arXiv preprint arXiv:2008.12073*, 2020.

[21] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A Versatile Diffusion Model for Audio Synthesis," *arXiv preprint arXiv:2009.09761*, 2020.

[22] J. Engel, C. Gu, A. Roberts *et al.*, "DDSP: Differentiable Digital Signal Processing," in *International Conference on Learning Representations*, 2019.

[23] X. Serra and J. Smith, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[24] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-To-Image Translation With Conditional Adversarial Networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[25] T. Hinz, M. Fisher, O. Wang, and S. Wermter, "Improved Techniques for Training Single-Image GANs," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1300–1309.

[26] S. Gur, S. Benaim, and L. Wolf, "Hierarchical Patch VAE-GAN: Generating Diverse Videos From a Single Sample," *arXiv preprint arXiv:2006.12226*, 2020.

[27] N. Granot, A. Shocher, B. Feinstein, S. Bagon, and M. Irani, "Drop the GAN: In Defense of Patches Nearest Neighbors as Single Image Generative Models," *arXiv preprint arXiv:2103.15545*, 2021.

[28] G. Greshler, T. R. Shaham, and T. Michaeli, "Catch-A-Waveform: Learning to Generate Audio from a Single Short Example," *arXiv preprint arXiv:2106.06426*, 2021.

[29] J. Nistal, S. Lattner, and G. Richard, "Comparing Representations for Audio Synthesis Using Generative Adversarial Networks," in *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 161–165.

[30] C. Gupta, P. Kamath, and L. Wyse, "Signal Representations for Synthesizing Audio Textures with Generative Adversarial Networks," *arXiv preprint arXiv:2103.07390*, 2021.

[31] Z. Průša, P. Balazs, and P. L. Søndergaard, "A Noniterative Method for Reconstruction of Phase From STFT Magnitude," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 5, pp. 1154–1164, 2017.

[32] D. Griffin and J. Lim, "Signal Estimation From Modified Short-Time Fourier Transform," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 32, no. 2, pp. 236–243, 1984.

[33] G. Le Vaillant, T. Dutoit, and S. Dekeyser, "Improving Synthesizer Programming From Variational Autoencoders Latent Space," 2021.

[34] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs," *arXiv preprint arXiv:1704.00028*, 2017.

[35] F. Font, G. Roma, and X. Serra, "Freesound Technical Demo," in *Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 411–412.

[36] J. van Doorn, A. Ly, M. Marsman, and E.-J. Wagenmakers, "Bayesian Rank-Based Hypothesis Testing for the Rank Sum Test, the Signed Rank Test, and Spearman's ρ," *Journal of Applied Statistics*, vol. 47, no. 16, pp. 2984–3006, 2020.

A QUANTUM MODEL FOR SOUND SYNTHESIS: QHOSYN

Rodney DuPlessis
UCSB
rodney@rodneyduplessis.com

ABSTRACT

The dynamic nature of quantum physical models make them well-suited to creative sound design and musical purposes through sonification and metaphor. These systems outline a new class of time-varying stochastic physical models. QHOSYN is a software synthesizer for simulating and sonifying the most fundamental of quantum mechanical models: the quantum harmonic oscillator. This basic model is used in physics to describe a plethora of more complex systems, and so it serves as a solid foundation for work in sonifying quantum systems. QHOSYN runs on Linux, Windows, and MacOS and allows composers and music researchers to explore the sonic potential in this physical model. In this paper, the software is introduced through a discussion of its background, an overview of the physical simulation implementation, a look at the visual interface, a description of the internal sound engine and how it can be extended through OSC, and finally a case study of using the software in composition.

1. INTRODUCTION

Dennis Gabor was the first to suggest a quantum notion of sound [1, 2]. Inspired by this, Iannis Xenakis developed a theory for music composition using sound grains [3]. Xenakis conceived of arranging these sound grains according to probabilistic laws derived from information theory and thermodynamics. Later, Curtis Roads greatly expanded the theory of microsound and granular synthesis [4]. Much of this work in formalizing a granular approach to sound focused on the novel idea of treating sound as a mass of particles. This has opened an enormous field of possibilities in sound analysis, synthesis, and composition [5]. Conversely to how treating an elementary particle as a wave was a paradigm shift for physicists in the early 20th century, treating the traditionally wave-like phenomenon of sound as a particle was a novel concept for musicians in the late-20th century.

The research presented here returns to Gabor's original meaning by treating sound as quantum particles in the most literal sense. These quantum particles, described by the mathematics of quantum mechanics, are well known to exhibit characteristics of both particles and waves, to defy causality, and to exist in superpositions of probability states. Using this kind of entity and the mathematics of quantum physics in sonification can lead to novel and compelling musical material [1].

The idea of using quantum physical simulations in music has seen fairly little exploration aside from Rajmil Fischmann's Erwin software [6] and a piece by Bob Sturm [7]. The Allosphere Research Group, led by JoAnn Kuchera-Morin, has done some of the most compelling work to date in sonifying and visualizing quantum processes [8].

QHOSYN (Quantum Harmonic Oscillator Synthesizer; pronounced as "cosine") was created to simulate a quantum harmonic oscillator for the purpose of exploring the possibilities of sonifying such a system. Like its counterpart in classical mechanics, the quantum harmonic oscillator is a fundamental model in quantum mechanics as it forms the basis of many systems [9]. Unlike the classical harmonic oscillator, the quantum harmonic oscillator describes a time-varying wavefunction, rather than a classical object. The ground state of this wave, like all quantum systems, is greater than zero. In other words, it cannot be totally static and must always be in some state of fluctuation. This dynamic nature suggests musical possibility.

A quantum harmonic oscillator can be described by its eigenstates, or possible energy levels. The wavefunction of the quantum harmonic oscillator is a linear combination (or superposition) of these eigenstates. The wavefunction is complex-valued and rotates in the complex plane (phase increments). Each eigenstate, then, can interfere based on their phase components in a superposed state. This creates a time-dependent morphing wave and it was this evolving wave pattern that inspired creation of QHOSYN.

QHOSYN can simulate the quantum harmonic oscillator in a superposition of up to 15 eigenstates. It is both a simulation and a tool for generating quantum-based sound. Several sonification strategies were explored (described in section 4). In the most direct sonification, the quantum harmonic oscillator itself is used as a waveform for wavetable synthesis. Beyond this is a stochastic strategy based on the time-varying probability distributions of the quantum harmonic oscillator's wavefunctions. Thus, this work is close in spirit to Xenakis' pioneering work incorporating stochastic theory into musical composition [3].

Copyright: © 2022 Rodney DuPlessis. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[1] This paper includes sound and video examples. If you click examples in the text, you will be taken to the corresponding media example online. If you are reading a paper copy of this document, or if your PDF reader does not support hyperlinks, you may navigate to the media examples manually in an internet browser at the following address: https://rodneyduplessis.com/media-examples/SMC2022

2. PHYSICS

To begin with the physical simulation, we first calculate the eigenbasis of the quantum harmonic oscillator. This will simplify the process of calculating new wavefunctions, as we need only multiply the wavefunction with each basis vector to obtain a new quantum harmonic oscillator.

The eigenbasis forming our Hilbert space is given by:

$$\psi(x,n) = \frac{1}{\sqrt{2^n n!}} \left(\frac{m\omega}{\pi\hbar}\right)^{1/4} e^{-\frac{m\omega x^2}{2\hbar}} H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right) \tag{1}$$

where n (an integer) is a unique eigenstate of the system and corresponds to the orthogonal axes in our Hilbert space, x is a position, m is the mass of the oscillator, ω is the angular frequency of the oscillator, and H_n are the physicist's Hermite polynomials, which orthogonalize our eigenstates:

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2} \tag{2}$$

Using nondimensionalization, we can normalize the units and simplify the problem. We can factor out the reduced Planck constant $\hbar$ as well as the mass and angular frequency of the oscillator, resulting in the simplified equation:

$$\psi(x,n) = \frac{\pi^{1/4}}{\sqrt{2^n n!}} \cdot e^{-(x^2/2)} \cdot H_n \tag{3}$$

This is the form that QHOSYN uses to determine the basis. QHOSYN calculates a 15-dimensional Hilbert space as the eigenbasis for our wavefunction, since we plan to allow up to the first 15 eigenstates to be simulated. This Hilbert space is calculated using the formula described above.

The `WaveFunction` class takes as arguments in its constructor an object of class `HilbertSpace` and an `std::function` that takes a double precision float and returns a double precision float. In its constructor, the `WaveFunction` then calculates an orthogonal basis projection from this `HilbertSpace`.

The wavefunction $\langle\psi|$ given by the user is projected onto the orthogonal vectors $|\phi\rangle$ of the Hilbert space. This entails a matrix multiplication for each energy eigenstate $|\phi\rangle$ that we wish to simulate:

$$\langle\psi_x|\phi_x\rangle = \psi_0\phi_0 + \psi_1\phi_1 + \psi_2\phi_2 + \cdots + \psi_n\phi_n \tag{4}$$

The `orthogonalBasisProjection()` method creates a lambda function for each dimension of the Hilbert space. This lambda function takes a float value, samples the Hilbert space in that basis, gets the complex conjugate of the sampled result, and returns the magnitude of that conjugate multiplied by a sample of the wavefunction. In pseudo-code this looks like:

```
float X =
conjugate(hilbertSpace[dimension][x])
* waveFunction[x]
```

This function is then integrated over the integration interval of -10 to 10 in each basis to find the coefficient for each eigenstate. For this, the CQUAD doubly-adaptive integration method from the GNU Scientific Library is used [10]. A normalization factor is then calculated to bring the resulting wavefunction into a usable amplitude range. The resulting wavefunction is a complex valued projection of the superimposed eigenstates. Specifically it is a discretized version of what would be a continuous function, in order to be computable. The spatial resolution of this discretization is 256. This number was chosen as it is a reasonable length to use as a waveform in wavetable synthesis or as a complex array in Fourier synthesis, and it is lean enough to enable fast calculations and broadcasting via OSC.

Once all of the above is finished, the wavefunction is now stored in memory. Every step of the simulation needs only to lookup the value for each position along the x axis at time step t in each orthogonal vector and add the results:

$$\langle\psi(x,t)| = \sum_{n=0}^{N} c_n \phi_{n,x} t\,(0.5+n)\,i \tag{5}$$

Time t is multiplied by the eigenstate number (energy level). This gives us the imaginary part of our complex number. The value of x at time t for each $|\phi\rangle$ is multiplied by the coefficient c_n for that vector derived from the orthogonal basis projection or given manually by the user. This gives us our wavefunction at every step of the simulation. To obtain the probability distribution, the wavefunction is simply squared.

QHOSYN also provides a means to "measure" the quantum harmonic oscillator. This measurement reveals the true position of the simulated particle represented by the quantum harmonic oscillator based on its probability distribution. A measurement like this would normally collapse the wavefunction of the quantum object through a process called decoherence. In QHOSYN, the effects of decoherence are deliberately ignored in order to maintain the simulation in a state of quantum superposition. This is a somewhat idealized system, similar to that of quantum computing researchers, who try to maintain qubits in a coherent state to preserve the pertinent quantum effects [11].

3. VISUAL INTERFACE

The visual interface of QHOSYN is comprised of the visualization of the quantum harmonic oscillator and several control panels that can be used to configure the simulation, the visualization and sonification, and manage data flow (see figure 1). The visual interface updates at 60 frames per second. User input is accepted via mouse and keyboard.

The visualization of the quantum harmonic oscillator is the dominant element of the interface. The wavefunction itself is represented by a red line. The probability distribution calculated from the wavefunction is a blue line. The three axes (x, y, z) are white lines and a grid is drawn in dark gray lines. If the probability noise band or inverse Fourier transform modes are activated, then the waveform

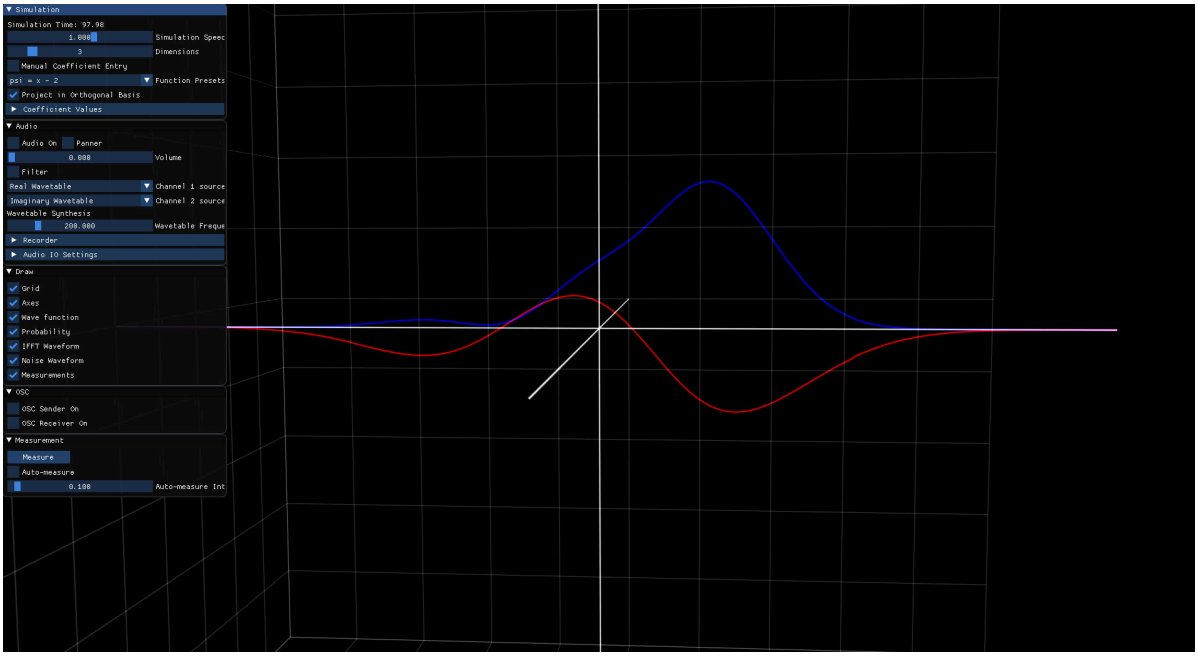


Figure 1. The visual interface of QHOSYN.

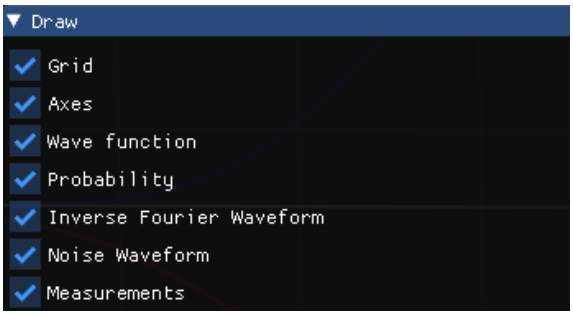


Figure 2. The Draw Panel.

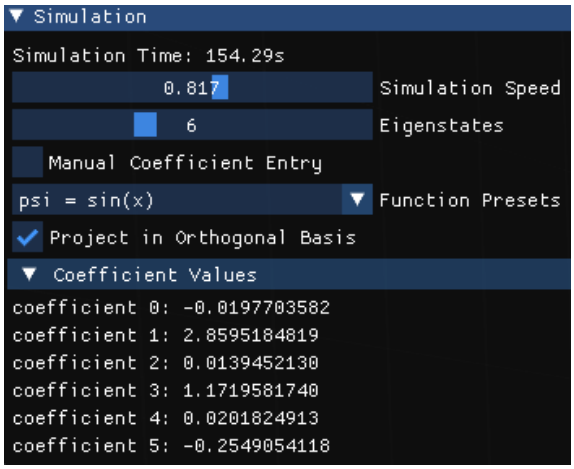


Figure 3. The Simulation Panel.

of those sonifications appear in yellow and cyan, respectively. Any of these visualizations can be activated or deactivated in the ”Draw” control panel(figure 2). This allows the user to customize the visualization according to their preference.

The wavefunction evolves over time, its phase rotating and, if the wavefunction is not in a stationary state, its shape changing. The speed of the simulation can be adjusted in the ”Simulation” panel(figure 3). The speed can be set between -5 and 5 and is a multiplier on the simulation speed. A speed of 0 will freeze the simulation.

In the simulation panel, the user can also change the number of eigenstates being simulated (between 1 and 15). The basis eigenfunction can be changed from the ”Function” dropdown menu. There are currently five presets representing the following eigenfunctions:

$$\begin{aligned} |\phi(x)\rangle &= sin(x) \\ |\phi(x)\rangle &= if(x = MaxEigenstate, 1, 0) \\ |\phi(x)\rangle &= rand(0, 1) \\ |\phi(x)\rangle &= x - 2 \\ |\phi(x)\rangle &= \frac{1}{x+1} \end{aligned} \tag{6}$$

If the user selects ”custom” from the Function dropdown menu, a text box will appear into which the user can input a c-style math function. QHOSYN can parse a wide array of arithmetic operators and trigonometric functions (the user can click the ”help” button for syntax hints). When the user presses enter after inputting a function, QHOSYN automatically parses the text input into a format that can be applied to the simulation.

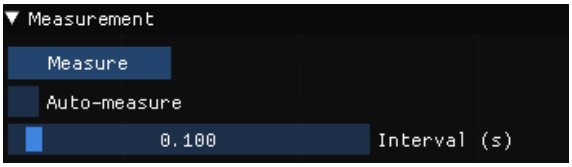


Figure 4. The Measurement Panel.

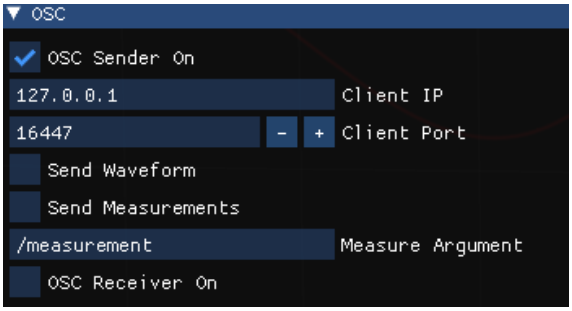


Figure 5. The OSC Panel

The Simulation panel also allows the user to set the coefficients of each eigenstate manually if the "Manual Coefficient Entry" checkbox is selected. The user can also choose to apply the selected function directly to the eigenstate coefficients without orthogonal basis projection by de-selecting the "Project in Orthogonal Basis" checkbox.

Measurements of the wavefunction can be taken by using the controls in the "Measurement" panel (figure 4). A single measurement can be taken by clicking the "Measure" button, or measurements can automatically be taken at a regular interval by activating the "Auto-measure" checkbox and selecting a measurement interval using the Interval slider. When taking measurements, the probability curve of the wavefunction provides a random discrete weighted distribution function. The result of this stochastic sampling is visually displayed on the interface by a white glowing point on the x-axis, indicating the measured position of the particle. This measurement can in turn be sent out via Open Sound Control (OSC) to another application, or control the panner within QHOSYN's sound engine.

The "OSC" panel provides controls for configuring the OSC sending and receiving capabilities of QHOSYN (figure 5). By selecting the "OSC Sender On" checkbox, the controls for OSC sending are revealed (the same for the "OSC Receiver On" checkbox). These controls allow the user to configure the IP address and Port over which to send the OSC messages, the argument to use to identify measurement messages from QHOSYN, whether to send measurement data, and whether to send the entire waveform (as an array).

The "Audio" panel contains the controls for QHOSYN's sound engine (figure 6). Audio can be toggled using the "Audio On" checkbox, and the volume of the audio output can be set with the "Volume" slider. The "Panner" checkbox activates the panner, which pans the sound according to the measurement results (auto-measure must be on). Both audio channels are panned independently. "Channel 1 Source" and "Channel 2 Source" dropdown

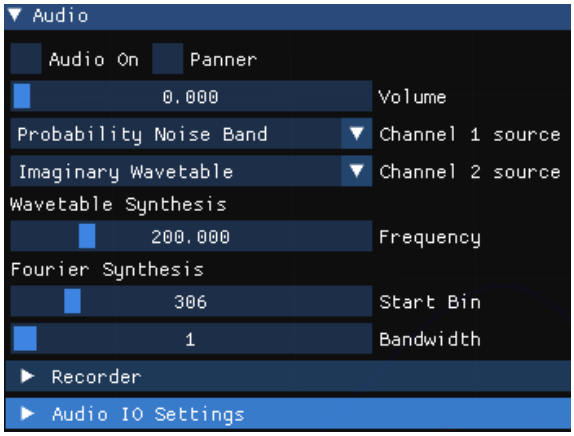


Figure 6. The Audio Panel

menus allow the user to select the sources for these channels from the following options: none, Real Wavetable, Imaginary Wavetable, Probability Wavetable, Probability Noise Band, and Inverse Fourier Transform. The "Frequency" slider controls the frequency of wavetable synthesis, and the "Start Bin" and "Bandwidth" sliders control the Fourier transform.

4. SOUND ENGINE

The internal synthesis engine of QHOSYN converts the quantum harmonic oscillator into sound in three ways: wavetable synthesis, inverse Fourier transform synthesis, and noise band synthesis. More possibilities emerge by pairing QHOSYN with an external application through OSC communication. In all cases, the sound is directly derived from the model of the quantum harmonic oscillator.

The wavetable synthesizer (see Video Example 1) is the most direct method of sonifying the wavefunction of the quantum harmonic oscillator. This method uses the oscillating shape of the wavefunction as an evolving waveform. QHOSYN reads through the waveform a number of times per second determined by the value of the Frequency slider set by the user. The wavetable can be derived from the real values of the wavefunction (y axis), the imaginary values (the z axis), or the probability distribution (the magnitude squared of the wavefunction). The wavetable reader always proceeds from left to right on the x axis. When the synthesizer read head reaches the end of the waveform, it wraps back around to the beginning. If the read head lands between sampled values of the wavefunction, it linearly interpolates between the preceding and succeeding sample. The waveform is updated 60 times per second, at the visual frame rate of QHOSYN.

The wavefunction is stored in memory as an array of 256 complex numbers for each time step. The inverse Fourier transform method of synthesizing the wavefunction treats these arrays as frames of a Short-Time Fourier Transform (STFT). An STFT breaks a signal into blocks and applies a Fast Fourier Transform algorithm to each block, returning a series of complex-valued arrays that represent slices

of the signal in time. These complex arrays provide information about the frequency, magnitude, and phase content of the signal. One can convert this STFT representation of the signal back into a time-domain signal through an Inverse Short-Time Fourier Transform (ISTFT) process. The inverse Fourier synthesis of QHOSYN takes advantage of the morphological similarity between the data of the wavefunction and the STFT. The sonic result resembles a band-limited impulse train with an evolving spectral envelope (see Video Example 2 and Sound Example 1).

An extension of this technique leads to the noise band synthesis method of QHOSYN. This method follows the same procedure of taking the IFFT of the wavefunction data, except that it randomizes the phase. First, the magnitude of each complex number is taken ($mag = \sqrt{im^2 + re^2}$). Then a random phase is calculated and the real part of the bin is calculated ($re = cos(phase) * mag$) and the imaginary part is calculated ($im = sin(phase) * mag$). The IFFT is then taken form this data. The result of this method is colored noise with a spectrum shaped by the probability distribution curve of the wavefunction, shifting and morphing in time (see Video Example 3 and Sound Example 2).

Beyond these synthesis algorithms, another sonic effect was derived from the wavefunction's stochastic nature: The panner. One of the most important purposes of the quantum harmonic oscillator model is to describe the probability of finding a quantum object in a particular eigenstate. In the simplest example, the eigenstate we are looking for is position. The particle has an indeterminate position, and the wavefunction tells us the probability of finding it in any given position. This leads to the idea of measuring the wavefunction by sampling the probability distribution to generate stochastic panning data. A key feature of this stochastic data is that it is not only based on a physical model, but this model describes a non-stationary probability distribution.

The panner can be applied to all of the synthesis methods described here (see Video Example 4). The panner relies on the measurement sampler, which returns a value from the weighted probability distribution of the wavefunction. Each of the two channels gets a different value. Regardless of which synthesis method is chosen, the sound will be panned in the stereo field according to the position measurement.

5. QHOSYN IN SITU

QHOSYN was used heavily in the composition of the author's fixed media electroacoustic work *Psi*.

There are many ways in which quantum harmonic oscillators can be used to generate sound and musical material. *Psi* makes use of at least six methods: the two Fourier methods (the IFFT method and the colored noise method) mentioned in the previous section, the wavetable synthesis method, quantum panning, and two granular paradigms.

The sound material of *Psi* consists of a high-quality recording of a Newton's Cradle, and over 90 unique sound files generated using QHOSYN. Some of the sound files use QHOSYN's internal synthesis, while others use QHOSYN's OSC data to control another application. The sounds generated with QHOSYN incurred very little post-processing, if any, so that the link to the quantum sound is preserved.

5.1 Wavetable Synthesis

The wavetable synthesis method of sonifying the wavefunction proved to be quite musically inspiring and was further enhanced through spatial effects using stochastic data generated by sampling the wavefunction's probability curve (see Sound Example 3).

It turns out that the most obvious aural effect of the wavetable synthesis is actually the shifting momentum of the particle, not the position. The Fourier transform of a time-varying signal gives us a frequency domain signal from which we can derive the magnitude of a frequency in the original signal. On the other hand, the Fourier transform of a position-varying wave function gives us a momentum curve, the magnitude of which gives us the probability distribution of possible momenta. Time is to frequency as position is to spatial frequency. The phase of a sound signal varies over time whereas the phase of a wavefunction varies over position. The derivative of phase with respect to time (the rate of change of phase over time) is equal to frequency. The derivative of phase with respect to position (the rate of change of phase over position) is equal to spatial frequency. Treating the wave function as a table for wavetable synthesis essentially converts the position axis to a time axis.

The spatial frequency of the Fourier transform becomes temporal frequency. Thus, the evolving spectrum we hear is analogous to the evolving momentum distribution over time. The spectrum is harmonic due to the quantized nature of the energy states of a particle. The energy states in superposition are integer multiples of the ground state,

5.2 Granular Synthesis

Granular synthesis is an apt framework for the sonification of quantum processes. Just as quantum physics explores the most infinitesimal objects in the universe, granular synthesis allows composers to create music using the smallest sonic particles. For the granular explorations of quantum physics in *Psi*, two software applications helped to extend the capabilities of QHOSYN: EmissionControl2, and Pulsar .

EmissionControl2 provided a broad field of possibilities to explore with its 15 parameters, unlimited sound file granulation, and flexible modulation scheme [12]. Measurements of the particle's position in QHOSYN controlled via OSC parameters of EmissionControl2. One of the most direct mappings was to apply these measurements to panning, so that spatial measurements applied to spatial effects in the sound.

Granular synthesis also afforded a metaphor with wave-particle duality. Sonic particles, after all, are actually waves, though they have finite bounds in time and space. Additionally, sound particles make up a Gestalt in granular synthesis that itself might be perceived as wave-like or particulate. Throughout Psi wave-particle duality is evoked by

sounds and grain streams in the frequency range at the edge of perception between rhythm and pitch (15-25 Hz). In this range, the lister sometimes perceives individual grains, and sometimes perceives a low frequency wave formed from the fusing of grains.

5.2.1 Pulsar Synthesis

Pulsar synthesis is a method of granular synthesis that emits grains at a regular interval. The pulsar is made up of a "pulsaret" followed by a silent interval. The composer can generate musical sounds by controlling the proportion of pulsaret to silence in the pulsar, the waveform of the pulsaret, the amplitude envelope, and the frequency of the pulsar stream [13]. The author created Pulsar in 2018 to explore the possibilities of this technique in Pure Data [14]. A major benefit of Pulsar in the case of composing for Psi was the ability to flexibly apply data to the parameters thanks to the Pure Data environment.

Pulsar synthesis presents an especially compelling analogy for the perception of the quantum mechanical system in question. With it's short, regular pulses, a sampling of the quantum oscillator applied to some of the parameters of pulsar synthesis quickly immerses the listener into the quantum world (see Sound Example 4). Two Pulsar objects were instantiated in Pure Data, and measurement data from QHOSYN was sent to each of them to control their panning. The routing alternated so that the two pulsars did not change position at the same time. The wavefunction of the quantum harmonic oscillator itself provided the waveform of the pulsars. The real part of the wavefunction went to one pulsaret waveform, and the imaginary part to the other (see figure **??**). Pulsar synthesis is commonly done with a static pulsaret waveform [13]. This method resulted in a novel form of pulsar synthesis with dynamic pulsaret waveforms that evolve smoothly in time.

5.2.2 3D QHO Granular Clouds

Three instances of QHOSYN can simulate a three-dimensional quantum harmonic oscillator, analogous to a real particle, taking advantage of the fact that the dimensional terms are separable in the three-dimensional Schrödinger equation [15]. For some of the sound material in *Psi*, such a configuration was used and the measurement data from all three were sent to EmissionControl2 (see Figure **??** and Video Example 5). The x-axis measurement results governed the stereo panning of grains. The y-axis measurement controlled the pitch of grains by modulating the Playback Rate parameter. The z-axis measurements affected the a sense of "depth" of the grains by modulating the Grain Duration and at the same time modulating the filter Resonance (higher z value resulted in longer and more filtered grains, lower z created shorter and less filtered grains).

Using this configuration, it was possible to create three-dimensional quantum granular clouds. This method approached a kind of sonification of a particle in 3D space measured at regular intervals. Under this correspondence, each grain represents the particle in the spatial location it is found each time it is measured. While each measurement represents the particle at a time and place, a collapsing of the positional potential of the particle, the sonic result is not a well-defined particle. The perception, with so many rapid measurements, is rather a Gestalt mass or cloud of sound that loses definition and takes on a fuzzy character. Thus, the resulting granular clouds contain traces of the quantum probability cloud.

By manually controlling other parameters such as Grain Rate, Asynchronicity, Intermittency, Filter Center, and Envelope Shape, a variety of quantum cloud textures were created. Naturally, the choice of which sound file to granulate had an impact on the quantum cloud texture as well. This sound material included some of the sound files generated from QHOSYN's internal sound engine for *Psi* as well as the recording of the Newton's Cradle that is featured in the piece.

5.3 Quantum Panning

One of the foundational musical ideas of *Psi* is a spatial one. The indeterminate position of quantum particles is among their most noteworthy qualities. QHOSYN's ability to take periodic measurements of its quantum harmonic oscillator simulation allows one to generate a stream of evolving stochastic panning data, imbuing any sound with this quantum quality of indeterminate position.

In *Psi*, this quantum panning effect is applied to most of the sound generated with the synthesis strategies discussed in this section. The rate of measurements taken becomes another synthesis parameter, controlling how frequently the panning is updated.

6. CONCLUSION

Sonification and musification of scientific data and frameworks creates opportunities and poses challenges for the composer. Physics in particular is perhaps the most accommodating of the translation into music of all the sciences. It may be because of the tangibility of the theoretical models, though that aspect does not apply to quantum physics. Certainly any dynamical time-varying theory will lend itself to sonification as music is a dynamic, time-varying art form. Some models in science are static descriptions of a system, and these models tend to lead to musical dead-ends. The most musical scientific models describe the way multiple objects or agents interact over time. From a model such as this, contrapuntal relationships, dynamic gestures, and shifting hierarchies can be derived that naturally propel music forward.

The quantum harmonic oscillator simulated by QHOSYN is such a model. It is also a basic and fundamental model of quantum mechanics that is used as a building block in more elaborate theories. With the initial version of QHOSYN, it was desirable to focus on this most fundamental model to lay the groundwork for more elaborate experiments to come. In a future version of QHOSYN further features and physical models will be explored, such as a collapse mechanic, so that taking a measurement can collapse the wavefunction, followed by dispersion. Other numerical solutions to the time-dependent Schrödinger equa-

tion may allow for real-time user interference in the wave function's evolution. These interferences could cause wave function collapse or trigger creation and annihilation operations (adding or removing energy eigenstates). Lastly, another direction for QHOSYN is to simulate coupled quantum oscillators that might approach a simulation of phonons or quantum fields. Beyond these plans for extending QHOSYN, it will be interesting to see the directions in which other researchers and musicians expand the quantum metaphor in sonification.

Research that extends the quantum harmonic oscillator model described here or that explores other quantum models could reveal a whole new class of time-varying stochastic musical forms. Certainly there is potential in exploring this on the quantum computing machines that are becoming more feasible today. Further possibilities may yet emerge as musicians begin to explore and play with the mathematics, data structures, and paradigms offered by quantum physics. To this end, I have made QHOSYN free and open-source so users can play with the software or probe, modify, and re-purpose the code-base. One can examine the code or download compiled and ready to run versions of the software for Linux, Windows, and MacOS at https://github.com/rodneydup/QHOSYN.

Acknowledgments

The author thanks the following people for inspiration, guidance, and advice that led to the design of QHOSYN: Curtis Roads, Clarence Barlow, João Pedro Oliveira, JoAnn Kuchera-Morin, Mason Hock, Kramer Elwell, Raphael Radna. Parts of this research were supported by a grant from the graduate division of UCSB.

7. REFERENCES

[1] D. Gabor, "Theory of communication," *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–457, 1946. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/ji-3-2.1946.0074

[2] ——, "Acoustical quanta and the theory of hearing," *Nature*, vol. 159, no. 4044, pp. 591–594, 1947. [Online]. Available: http://www.nature.com/articles/159591a0

[3] I. Xenakis, *Formalized Music: Thought and Mathematics in Composition*. Pendragon Press, 1992, google-Books-ID: y6lL3I0vmMwC.

[4] C. Roads, *Microsound*. MIT Press, 2002.

[5] B. L. Sturm, C. Roads, A. McLeran, and J. J. Shynk, "Analysis, visualization, and transformation of audio signals using dictionary-based methods †," *Journal of New Music Research*, vol. 38, no. 4, pp. 325–341, 2009. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/09298210903171178

[6] R. Fischman, "Clouds, pyramids, and diamonds: Applying Schrödinger's equation to granular synthesis and compositional structure," *Computer Music Journal*, vol. 27, no. 2, pp. 47–69, 2003. [Online]. Available: https://www.mitpressjournals.org/doi/abs/10.1162/014892603322022664

[7] B. L. Sturm, "Composing for an ensemble of atoms: the metamorphosis of scientific experiment into music," *Organised Sound*, vol. 6, no. 2, pp. 131–145, 2001. [Online]. Available: https://www.cambridge.org/core/product/identifier/S1355771801002102/type/journal_article

[8] L. J. Putnam, J. Kuchera-Morin, and L. Peliti, "Studies in composing hydrogen atom wavefunctions," *Leonardo*, vol. 48, no. 2, pp. 158–166, 2015. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/LEON_a_00912

[9] S. J. Ling, J. Sanny, and W. Moebs, "The quantum harmonic oscillator," 2016, book Title: University Physics Volume 3 Publisher: OpenStax. [Online]. Available: https://opentextbc.ca/universityphysicsv3openstax/chapter/the-quantum-harmonic-oscillator/

[10] The GSL Team. Numerical integration — GSL 2.7 documentation. [Online]. Available: https://www.gnu.org/software/gsl/doc/html/integration.html#cquad-doubly-adaptive-integration

[11] K. McCormick. Decoherence is a problem for quantum computing, but ... [Online]. Available: https://blogs.scientificamerican.com/observations/decoherence-is-a-problem-for-quantum-computing-but/

[12] C. Roads, J. Kilgore, and R. DuPlessis, "Architecture for real-time granular synthesis: EmissionControl2," *Manuscript submitted for publication*, p. 27, 2021.

[13] C. Roads, "Sound composition with pulsars," *Journal of the Audio Engineering Society*, vol. 49, no. 3, pp. 134–147, 2001, publisher: Audio Engineering Society. [Online]. Available: https://www.aes.org/e-lib/browse.cfm?elib=10198

[14] R. DuPlessis, "pulsar˜," 2021-10-11, original-date: 2020-04-05T04:24:58Z. [Online]. Available: https://github.com/rodneydup/pd-pulsar

[15] D. M. Fradkin, "Three-dimensional isotropic harmonic oscillator and SU3," *American Journal of Physics*, vol. 33, no. 3, pp. 207–211, 1965. [Online]. Available: http://aapt.scitation.org/doi/10.1119/1.1971373

Personalizing AI for Co-Creative Music Composition from Melody to Structure

Ken Déguernel[1] **Mathieu Giraud**[1] **Richard Groult**[2,3] **Sébastien Gulluni**
[1]Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille
[2]Normandie Univ., UNIROUEN, LITIS, F-76000 Rouen
[3]Univ. Picardie Jules-Verne, MIS, F-80000 Amiens
contact@algomus.fr

ABSTRACT

Co-creativity is a unique artistic situation where human interact with computer, and raises challenges on interaction, steerability, and personalization. We present a new co-creative music composition approach that we used for our participation in the "AI Song Contest 2021", an international music contest involving artificial intelligence (AI). We *personalize* the artificial creativity methods to adapt to the needs and expectations of a composer. Interactions between the composer and different AI methods occurred throughout the whole composition process, for the generation of melodies, chord progressions, global structure, and textural variations, both through *data sharing* for machine learning based AI and through *knowledge sharing* for rule-based AI. We describe these AI methods and how the composer interacted with them: The personalization of the AI methods enabled the composer to explore new musical territories while keeping their original style, with AI music generation which *"sounds like it has been generated for him"*. The song "The last moment before you fly" ranked 3rd place in this contest, the jury underlining the "personal feel" of the song. We discuss here how these methods open the way to new co-creative approaches, both using AI and personalization.

1. INTRODUCTION

1.1 Music Composition and Co-creativity

Music assisted composition developed throughout the years, employing the new scientific advances: In the 1950s, automated compositions mainly used first-order Markov chains. Fred and Carolyn Attneave analyzed and generated Western-style melodies described as "convincing" [1], whereas Pinkerton generated melodies from the analysis of 39 children's songs [2]. Probabilistic methods were expanded through several studies [3, 4]. Methods based on n-grams were used to predict (using entropy measures) and generate melodies in the style of Gregorian music or in the style of Bach [5]. Models based on linguistics were also used, using sets of rules and/or grammars to describe composition processes [6]. Evolutionary computation methods for music generation have been developed for the generation of melodies, drum tracks, or for live-coding applications [7]. Neural networks have been trained on Bach and traditional European melodies to extract stylistic regularities and to generate new melo'-dies [8]. Today, most machine learning methods use neural networks and deep learning techniques are used extensively for music generation [9–11].

Herremans et al. did a thorough review of music generation techniques [12]. Most of them focus on the generation of harmony (sequences of chords) and of melody (sequences of notes). One of the key challenges in music generation is to be able to generate structured music with long-term correlations [12]. Several strategies have been employed in order to integrate some long term structure in melody generation, for instance by using a combination of short- and long-term models [13], structural constraints [14, 15], or, with Markov models and their extensions, by modeling a prior hierarchical knowledge [16, 17]. Another strategy is to generate elements of a *narrative* providing a feel of long-term structure. For instance, Hyperscore [18] provides a graphical interface for computer-assisted composition to edit and to visualize musical structures from low- to high-level features (melodic shape, harmonic tension, etc.), and MorpheuS [19] uses a tension profile, designed by a user or computed from a template, as a constraint for music generation.

Another approach to music generation is to consider the AI models not as an independent agent, but rather as part of an interactive practice between human and machine, creating new dynamics of interactions through a *co-creative* process [20,21]. Co-improvisation systems such as DYCI2 [22] are based on real-time interactions between human musicians and a machine: Each performer (human or machine) listens to the music produced by the other and responds appropriately, bringing on the musical discourse in a novel way each time. Co-creativity can be used also in the context of a composition process either live through human-machine interactive systems [23, 24] or offline, in a *generate-then-select* fashion where the AI system generates a set of musical fragments, one of which will eventually be selected and possibly re-shaped by the composer to meet a specific musical need. For example, Ghisi proposes a method where a user chooses among fragments generated by a LSTM neural network [25], and Daren Banarsë composes folk music from melodies generated with a recurrent neural network [26]. Seeing the computer/AI system as a *colleague*, or even as a *co-creative colleague* who *sug-*

Copyright: © 2022 Ken Déguernel et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

gest ideas, was studied by Lubart [27], Kantosalo and Jordanous [28], and Micchi et al. [29].

1.2 Motivations and Contributions

In this article, we present the co-creative music composition process that served as the artistic ground of our collaborative submission to the AI Song Contest 2021. Our team consisted of a Music Information Retrieval (MIR) researcher team (KD, MG, RG), part of which already participated in the AI Song Contest 2020 [29], with diverse knowledge and different AI methods, from music analysis to music generation, and a professional composer (SG) who is also a sound designer.

Co-creativity raises challenges in interaction, steerability, and personalization [30]. Our main goal was here both to use existing tools from the field of automatic music generation, and to develop new techniques to further experiment with a *personalized co-creative approach* by maintaining a constant interaction between composer and AI methods. For this composition, we focused on:

- an exploration of a co-creative expression, through a collaboration **involving people with distinct roles**. The collaboration is between a composer and a MIR team equipped with different knowledge and AI.
- a composition that goes beyond a simple "*generate-then-select*" approach. We wanted to **involve the composer in the creative process right at the beginning of each generative process**. The first compositional gesture for each component is given by the composer through knowledge sharing for knowledge-based AI, and through the creation of a training corpus using the own data of the composer (as well as their influence) for machine-learning AI.
- the **use of AI methods throughout the whole pre-production compositional process**, on multiple levels of the compositional process, including the "big picture" of the song. That is to generate chords and melodies, but also methods to explore co-creativity for generating long-term structures and variations of compositional processes (e.g. melodic or rhythmic variations, timbral curves, etc.).

In Section 2, we present the methods and the different models we used and developed for the different needs and interactions that the MIR team wanted with the composer. In Section 3, we present the results, open up a discussion about our thoughts (especially the ones of the composer), as well as the contests' jury opinions, about the end results and the composition process.

2. METHOD

2.1 Approach

Generating structured music, especially involving long-term correlations between elements, is a key challenge [12, 31–34]. We decided to tackle this problem by focusing on a composition process with structural *sections*. First, harmonic progressions were generated by the AI models and selected by the composer for several structural sections. Second, melodies were generated upon the selected chord progressions for each section. Then, song structures were generated, organizing the order and repetitions of the different sections. And finally, *arrangement layouts* were generated on the structure with propositions of variations of timbre, rhythm, and melody.

We combined three types of "intelligence":

- the *human composer*, with their own compositional gesture, and music knowledge,
- *data-based AI* using machine learning for the generation of chords, and melodic contents,
- *rule-based AI* with some generative algorithms, for the generation of structures and variation templates.

Although the MIR team chose and developed the AI methods used, the interactions between the AI models and the composer were personalized through communication between them. Communication between the composer and the AI models was necessary at the initial step of each process (Figure 1). For the machine learning oriented AI, after initial knowledge sharing, that communication was *through data sharing*. For the generative algorithms, communication between the composer and the AI models was done *through knowledge sharing*, iterated and refined.

The methods presented here were developed during a 1-month timeframe, with a continuous discussion process among all the members of the team. The following paragraphs detail this composition process, focusing on personalization, for chords (Section 2.2), melody (2.3), structure (2.4), and finally variations (2.5). The music production is briefly explained in Section 2.6 but did not involve AI personalization.

2.2 Chords

Our first step was to create chord progressions for several sections of the composition. To personalize the generation, the first compositional gesture was done by the composer who selected a corpus of data mixing some of his own compositions as well as some of his influences (Ambient Electronica).

To generate new chord progressions, in the style of the given corpus, we used a factor oracle [35], a simple machine-learning AI which is at the core of the co-improvisation software developed at Ircam (OMax [36], SoMax [37], ImproteK [22], DYCI2 [38]). The factor oracle learns music content, here chord progressions, by connecting places in the memory which share a similar context. It can be considered as a variable order Markov model where each state has the maximum order possible in connection to the past, and can be used to generate new musical content whilst keeping the learned musical style. The main advantage of using a factor oracle for our approach is its capacity to generate music in a given style from a limited amount of data (compared to neural-network based methods). This is important for the personalization aspect of our approach as it would be difficult for a composer to

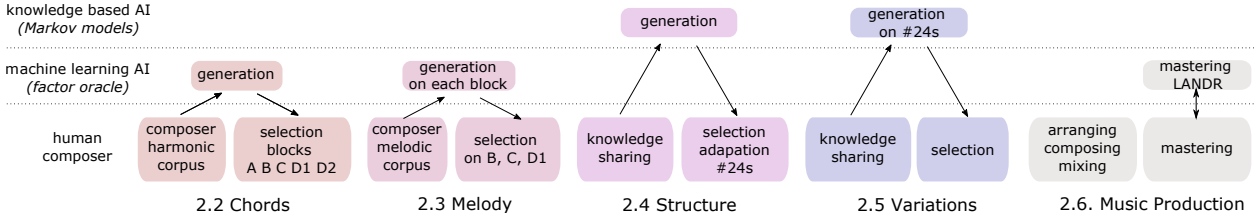


Figure 1. Workflow of our structure-based composition process and interactions between the composer and AI.

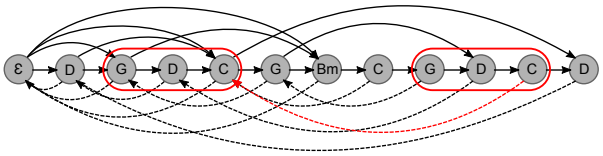


Figure 2. Factor oracle created from the chord progression `D G D C G Bm C G D C D`. Curved arrows at the upper half are used during the construction of the oracle, but are not used after anymore. Suffix links (dashed arrows) connect each state to the past states sharing the longest common context. The two states connected by the bold suffix link share the common context `G D C`. A possible generation by navigating the oracle is `D G D C` * `D . G`, taking this suffix link towards the future (*), then, after the last state `D`, taking a suffix link towards the past. This generation was not present in the original chord progression but nevertheless share significant sub-progressions with it.

provide a substantial amount of data representative of the music they are trying to compose. Moreover, the main limitation of the generative power of the factor oracle, i.e. the lack of long-term structure, is alleviated by the fact that the global structure is composed later using another method (Section 2.4).

The composer provided a corpus of 21 chord sequences (5 from his own composition and 16 from his influences) totaling 498 chords, including 49 different chords. Chords qualities include major/minor/7th chords (major, minor and dominant), but also suspended chords and slash chords. Chords were not transposed. We trained a factor oracle on these chord sequences (Figure 2). Generation then followed the heuristics proposed in [36], ie. the ability to jump both *forward and backward* in the memory, a *continuity factor* (3 chords must be generated before doing another jump in the memory), and a *taboo list* to avoid too many repetitions. The MIR team generated 50 new chord progressions of 16 bars each, and the composer created 5 sections, A, B, C, D1, and D2 from a selection of them (Figure 3).

The composer said:

> *"I started from the generated progressions that I truncated and split, selecting what pleased and surprised me. [...] Parts A and B are the beginning of the generations #37 and #38 that I truncated. The sequence from C to D1 is from the same generated chord sequences #41/#42 that I split into 2 parts (same for C and D2).*

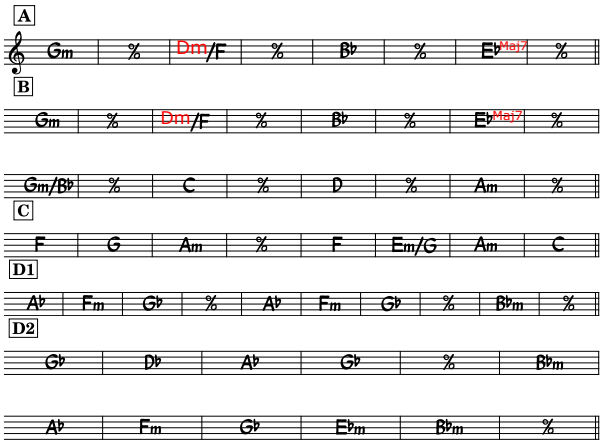



Figure 3. Chord progressions selected for each of the five sections. Red elements indicate further modifications chord made by the composer during the final touches of the composition process.

> *These parts sounded good in isolation but there was a missing link to go from B to C so I transposed C, D1 and D2 by a semitone to smooth the transition."*

The composer eventually expanded some chords to create a more modal feel.

2.3 Melodic content

Once the chord progressions for each section were selected, the next step was to generate melodies to go on top of them. Once again, to personalize the composition, the composer selected a corpus of 14 melodies for a total of 767 notes from his own music and inspirations. This corpus contained melodic and harmonic information.

A factor oracle was trained on this corpus, but this time the oracle was informed by interpolated probabilistic models [39] also trained on that same corpus. The factor oracle is created solely on melody, acting as a set of constraints on the probabilistic models which are representing the knowledge of the system about the melodic consistency, the chord-melody relationship and the balance between the two. The melodic consistency was modeled by a tri-gram: $P(M_t|M_{t-1}, M_{t-2})$ (where M_t is the note played at time t), and the chord-melody relationship was represented by a vertical model $P(M_t|C_t)$ (where C_t is the chord at time t). The two models were then interpolated with log-linear interpolation [40] and using

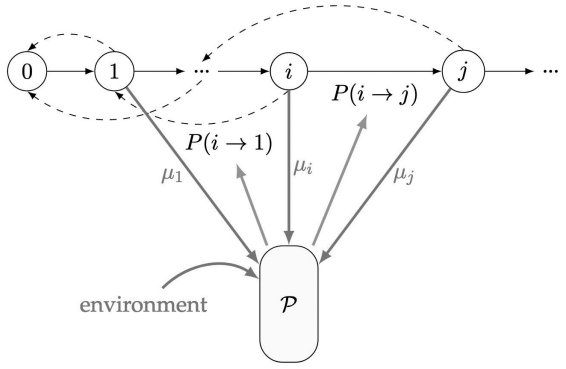


Figure 4. A probabilistic model linking melodies and chords is used to provide probability transitions towards each reachable states [39]. In our case, the factor oracle is built on melody, (μ_i corresponds to the melodic content in state i), whilst the environment is the chord progressions for each sections of the composition. At each step of the navigation, the transition probability for going for current state i to state j ($P(i \to j)$) is computed from the probabilistic models $P(M_t|M_{t-1}, M_{t-2})$ and $P(M_t|C_t)$, and then normalized according to the reachable states.

back-off smoothing [41]. Figure 4 shows a representation of one step in the navigation of a factor oracle informed by a probabilistic model. The probabilistic oracle proposed melodies for each of the chord progressions (taking into account melody and harmony).

> *"I used the oracle outputs as alternative suggestions for melodic direction. So I used the note sequences from the oracle and adapted them rhythmically to fit the song."*

Melodic contents have been used for the main melody but also to create background accompaniments for each section. In the end, not all selected melodies (either for the main melody or for the accompaniments) were actually used for each section in the composition: some sections working better with a single melody, or accompaniment parts appearing later creating some variational changes.

2.4 Structure

The third step of our composition process was to create a global structure for the piece, that is to decide how to combine the different sections with their harmonic and melodic elements. Candidate structures were generated by a first-order Markov model with additional constraints such as "do not output C before B already appeared". The constraints are created and the transition probabilities of the model are adapted to better emulate the style and intention of the composer in order to personalize the results. They are decided through knowledge sharing during conversations with the composer, with a refinement process: some structures are generated, the composer then assessed the quality of the generation, identify things that should occur more or less (or even not at all), new constraints are added and the probability of transitions are refined. For instance, the composer wanted to ensure that the initial A was repeated at least twice to "anchor the key" of the piece.

This process was repeated several times until most of the generated structures were satisfactory. The composer then selected one of them (Figure 5, top):

> *"I first selected structure 23 (that included D2 at the end), but, after three days of work, it did not work. I then tried to work with structure 24 – with another end – but finally, I decided to make it even more simple (looping on C), keeping a 100 BPM to fit the 4-minute song constraint. As with traditional composition, with AI, the eraser is the best tool!"*

.

2.5 Variations

Many experiments in AI (co-)creativity generate melody and/or chords. However, the arrangement plan is a key feature of a song. To add some co-creativity in the arrangement of the piece, the composer proposed a simple taxonomy of variations on rhythm, melody, and timbre/texture from his own conceptualization of his composition process (Figure 6).

A generative procedure based on a random selection of variations, upon generative curves, generates where and how variations are to be applied. For each musical dimension identified in the composer's taxonomy (rhythm, melody, timbre/texture), a curve is drawn out of a set of hand-crafted curves (decided through information sharing during discussions with the composer). The curve defines how many variations are being randomly selected out of the taxonomy of variations for each section of the piece. These curves were hand-crafted to have more variations towards the end of the song (Figure 7). Moreover, a dark/light curve was also generated by a similar procedure.

Twenty variations profiles were generated on structure 24s and the composer chose the variation profile 24s-02, and selected some of the variations inside this profile (Figure 5, bottom):

> *"I used the variations proposed by the AI to motivate me to give movement to the track while leaving me the possibility to improve the proposed solutions. Altogether, the structure and variation suggestion tools thus helped me to think outside the box, as for example, the timber/texture curve was not what I was thinking for."*

The composer chose how to implement each variation (which instruments and/or musical texture apply the different variations).

2.6 Final Touches, Music Arrangement, and Production

The composer created and produced the song given the harmony, the melody of each section, and the structure with

```
                structure 23:    AAA B C A C  D1 B C D2 A
                 structure 24:     AA B C A C  D1 B C D1
          ==> final structure 24s:     A B C A CC D1 B CC

## Structure with varations 24s-02
(Timb-0 A)
 Timb-0 A    (T-subst)
 Timb-1 B
 Timb-1 C
 Timb-2 A    (*R-dense*,M-trans)                [Sixteenth pulse]
 Timb-2 CC   (*M-irreg*,T-subst)                [Strings, C second time]
 Timb-0 D1   (T-far)
 Timb-0 B    (*M-ornem*,M-trans)                [Ornemented piano patterns]
 Timb-1 C    (*R-dense*,M-ornem,M-trans)        [Sixteenth pulse]
 Timb-2 (C)  (R-light,M-irreg,M-trans,*T-far*)  [Fade-out]
```

Figure 5. Final structure (top) with variations (bottom). The system proposes a timbre for each section from the structure as well as a series of potential variations for them. The starred variations were the one selected by the composer and the right side shows how this variation is implemented in the composition.

```
Rhythm R-dense      Densification
       R-light      Lightening
       R-frag       Conversion, fragmentation

Melody M-trans      Diatonic transposition
       M-irreg      Irregular copy
       M-ornem      Ornamentation, appogiaturas

Timbre, Texture
       T-far        Distancing
       T-near       Rapprochement
       T-susbt      Substitution
       T0..T1..T2   Dark..Light
```

Figure 6. Variation taxonomy. The initial vocabulary also containted harmonic variations, but the composer eventually decided that the chords had already enough such variation.

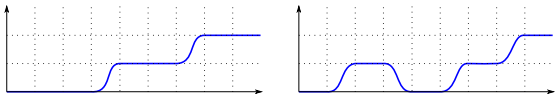


Figure 7. Variation curves. The left curve makes variations increase along the song. The right curve further pauses variations before the half of the song. Such curves were randomly selected, then stretched, to generate the variations on Figure 5.

variations presented above. Although some AI was also used during the music production part (after the composition), it did not involve AI personalization. The composer, being also sound designer and music producer, took great care in selecting/crafting instruments and textures in a minimalist "Ambient Electronica" style, combining synthesizer and drums with classical instruments like a cello and a string section. The main melody is played by a cello, while chords are presented with arpeggios played by a synthesizer. The densification and ornamentation are achieved by adding more complex accompaniment on strings or in the background, or by adding another layer of synthesizer playing the harmonic arpeggios with a new rhythm.

The mastering was done with the software LANDR that proposes online mastering services via an AI algorithm. Since the spectral shape of the song is not very common, AI based mastering fails to propose an adapted mastering

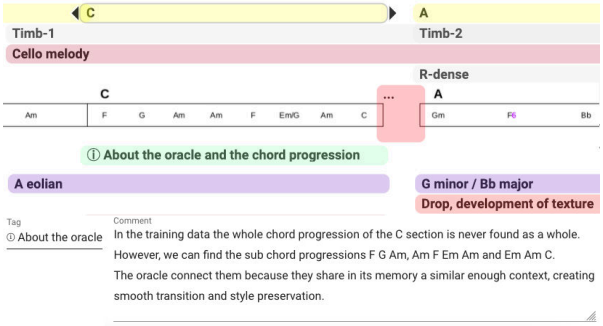


Figure 8. Interactive annotation of the song on Dezrann (`dezrann.net`). The top lines use the sections and variations features described on Figure 5. Selecting some labels, such as here "About the oracle", provides more information.

equalization so the composer decided to apply a slight mastering equalization before LANDR [42].

3. RESULTS AND DISCUSSION

In this section, we focus on how AI impacted the composition process for the different musical dimensions and how the personalized aspect of the AI mattered.

3.1 Song Availibility and Reception

The song "The Last Moment Before You Fly" is available online on various platforms. We also annotated the song, both for its global structure as well as for specific points describing some of the decisions taken during the compositional processes, on the Dezrann platform [1] (Figure 8).

To remain as transparent as possible regarding the impact of the AI on the song, we released the code that we used as well as some outputs of the generative process that has been used for the composition, under GNU GPLv.3 and Creative Commons CC-BY-SA 4.0 respectively. [2] Note that, as requested by the composer, we are not releasing the training corpora for the chords and melodic content, as it contains copyrighted material and is considered to be

[1] Links available from `www.algomus.fr/before-you-fly`
[2] `www.algomus.fr/code`

part of the composer's identity.

The song was awarded 3rd place at the AI Song Contest 2021 out of 38 entries. Mark Simos, a member of the jury of this contest, songwriter, said in the jury session:

> *"What was really unique about that project is the way that the composer got a chance to train the data on him specific individual artistic work, so he felt like he were actually having a kind of conversation with him own sort of stylistic corpus, which gave it a really kind of personal feel."*

3.2 Low-tech AI and Co-creativity

Indeed, we tried to follow the composer's creative process as much as possible for the AI generation of the different musical elements, and also to respond to him needs and expectation from his first time creating a piece with an AI system:

> *"My goal for this project was to use AI algorithms to find new musical ideas and provoke inspiration while staying within a predefined style. We employed a co-creative approach in which the AI methods and the human fuel each other's creativity in an iterative process."*

Our models were purposefully *"low-tech AI"* (low need of computing power/data), to ease the exchange and enhance the place of the composer in the generative process (being able to generate musical elements with a tailored-made but a limited amount of data). This way, we experiment more on co-creativity while involving AI in parts of the composition process that are traditionally human-made (arrangement). Moreover, this choice also complements the *minimalist vibes* of the composition, that the composer wanted.

For all layers of the composition, the composer was the one doing the first compositional gestures through knowledge and/or data sharing, enabling a true personalization of the AI models to him needs, and was also doing the final compositional gestures and had the final say.

3.3 A Modular Co-creative Experience

We started by creating blocks of chord progressions and melodies in order to decide the atmosphere and direction for the piece. The blocks were then arranged in a generated structure. This generation was experimental, and the probability curves were hand-crafted. We would definitely like to train more flexible statistical models on actual data – but this would require having such structural data that is nonexistent for now. Altogether, chords, melody, structure, and variations were perceived as stimulation by the composer. Even though some elements for them felt something unusual, he found the generation inspiring and enabled them to explore ideas he would not have had without him interaction with the AI models:

> *"The first use of the AI tools was for inspiration and to begin the creation process. It allowed me to expand my musical vocabulary and proposed some chord changes and melodic directions I would not have thought of without such assistance. When I was selecting the chord progressions there were a lot of options and it was difficult to choose. So, I decided to choose the progressions that moved me the most. I found that emotion that I wanted to capture and then I tried to make it grow through instrumentation and production. [...] I don't think I would have come up with these ideas without using these tools and I can say that this song is an artifact from my interaction with these algorithms."*

The MIR team was impressed by how fast the composer did appropriate the generative models with a clear insight on their inner working, which we think was possible by the "low-tech" nature of our methods. This facilitated a lot more in-depth discussion about the composition process, and helped with knowledge-sharing, in order to turn musical ideas into concrete computer model improvements and changes of model parameters.

Regarding the personalization of the AI, the methods using data for training presented here have a smaller generative power than some neural-network based methods from the state-of-the-art, such as [31, 43] which use much more data. However, the fact that factor oracle based method work on small corpora has an important impact on co-creativity. Creating a corpus large enough for neural network based methods would be much more difficult and time-consuming for a composer, and the overall content of the corpus would have to be more generic. Working on a smaller corpus is much more manageable for a composer and enables them to pinpoint more clearly the specific influences and style they want for their composition, which impacts directly their involvement in the human-AI interaction:

> *"Unlike the usual methods that use large generic corpora, the methods we employed used only my own data and influences to stay within a well-defined Ambient Electronica style. [...] Creating a corpus to train an AI with my own composition and influences makes it feel very personal. It was the first bricks of my own world that I would like to carry on contributing to by adding more stuff, piece by piece for future projects."*

With more time, more data would have been used, and the exploration of AI generation would have been longer by exploring more directions. The composer said:

> *"The result sounds less generic than some of the things I've heard from deep models. [...] It sounds like it has been generated for me."*

As said above, this aspect was also noted by members of the jury of the contest who were not part of the process.

3.4 Towards Autonomous Co-creation?

The method presented here for the generation of structure and variations shares similarities with MorpheuS [19]. In

both cases, profile curves are designed by the composer in order to guide the elements of the overall narrative of the composition. In our methodology, however, the composer define and describe their own vocabulary to describe their own approach to variation when composing. Therefore, the AI system is able to propose variations, communicating with the composer with their own taxonomy. On the one hand, this provides the composer with a better understanding of what the AI models are proposing. And on the other hand, this enables the composer to be more specific on which variational elements they want to consider in their composition. The autonomy is here shared between the machine and the human, who keeps such an autonomy "in the intentional sense" [44].

But could composers work by themselves, without any connection with a computer music team? A current limitation for the knowledge-based method is indeed the lack of a user interface, meaning that the MIR team was necessary to do the transition to interpret the needs of the composer and to implement them. Future work could include implementing such a user interface, in which the composer could describe their vocabulary, as well as a set of constraint and variation curves.

Here the MIR team and the composer worked in close interactions and the workflow evolved somewhat organically throughout the 1-month timeframe. More generally, we believe that this experience can be generalized and this workflow could be used again for a completely different project or with another composer. The different models used are agnostic of the data or of the knowledge, letting the composer express their intention and communicate with the AI models, through different data, and through new expressive vocabularies. However, we also felt that the interaction between humans, here the composer and the MIR team, was also central to the project, giving the composer a new perspective on their work while giving the MIR team new musical insights. As such, co-creation with AI also fosters creativity through new human relations.

4. REFERENCES

[1] C. Ariza, "An open design for computer-aided algorithmic music composition : athenaCL," Ph.D. dissertation, New York University, 2005.

[2] R. C. Pinkerton, "Information theory and melody," *Scientifc American*, vol. 194, no. 2, pp. 77–86, 1956.

[3] M. A. Hall and L. Smith, "A computer model of blues music and its evaluation," *Journal of the Acoustical Society of America*, vol. 100, no. 2, pp. 1163–1167, 1996.

[4] D. Conklin and J. Cleary, "Modelling and generating music using multiple viewpoints," in *1st workshop on AI and Music*, 1988, pp. 125–137.

[5] D. Conklin, "Music generation from statistical models," in *AISB Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, 2003, pp. 30–35.

[6] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. MIT Press, 1983.

[7] R. Loughran and M. O'Neill, "Evolutionary music: applying evolutionary computation to the art of creating music," *Genetic Programming and Evolvable Machines*, vol. 21, no. 1, pp. 55–85, 2020.

[8] M. C. Mozer, "Neural network music composition by prediction : exploring the benefits of psychoacoustic constraints and multi-scale processing," *Connection Science*, vol. 6, no. 2–3, pp. 247–280, 1994.

[9] G. Bickerman, S. Bosley, P. Swire, and R. M. Keller, "Learning to create jazz melodies using deep belief nets," 2010, pp. 228–236.

[10] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation." in *International Society for Music Information Retrieval Conference (ISMIR 2017)*, 2017, pp. 324–331. [Online]. Available: https://doi.org/10.5281/zenodo.1415990

[11] S. Ji, J. Luo, and X. Yang, "A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions," *arXiv:2011.06801*, 2020.

[12] D. Herremans, C.-H. Chuan, and E. Chew, "A functional taxonomy of music generation systems," *ACM Computing Surveys*, vol. 50, no. 5, pp. 1–30, 2017.

[13] M. Pearce, M. Ruiz, S. Kapasi, G. Wiggins, and J. Bhattacharya, "Unsupervised statistical learning underpins computational, behavioural and neural manifestations of musical expectations," *NeuroImage*, vol. 50, no. 1, pp. 302–313, 2010.

[14] A. Donzé, R. Valle, I. Akkaya, S. Libkind, S. A. Seshia, and D. Wessel, "Machine improvisation with formal specifications," in *International Computer Music Conference (ICMC 2014)*, 2014, pp. 1277–1284.

[15] T. Tanaka, B. Bemman, and D. Meredith, "Constraint programming formulation of the problem of generating milton babbitt's all-partition arrays," in *22nd International Conference on Principles and Practice of Constraint Programming*, 2016.

[16] K. Déguernel, E. Vincent, J. Nika, G. Assayag, and K. Smaïli, "Learning of hierarchical temporal structures for guided improvisation," *Computer Music Journal*, vol. 43, no. 2, 2019.

[17] N. Carvalho and G. Bernardes, "SyVMO: Synchronous variable markov oracle for modeling and predicting multi-part musical structures," in *International Conference on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART 2021)*, 2021.

[18] M. Farbood, H. Kaufman, and K. Jennings, "Composing with hyperscore: An intuitive interface for visualizing musical structure," in *International Computer Music Conference (ICMC 2007)*, 2007.

[19] D. Herremans and E. Chew, "Morpheus: Automatic music generation with recurrent pattern constraints and tension profiles," in *IEEE TENCON*, 2016.

[20] G. Assayag, "Creative symbolic interaction," in *Sound and Music Computing Conference (SMC 2014)*, 2014.

[21] P. Esling and N. Devis, "Creativity in the era of artificial intelligence," *arXiv:2008.05959*, 2020.

[22] J. Nika, M. Chemillier, and G. Assayag, "Improtek: Introducing scenarios into human-computer music improvisation," *Comput. Entertain.*, vol. 14, no. 2, Jan. 2017. [Online]. Available: https://doi.org/10.1145/3022635

[23] J.-M. Fernández, T. Köppel, G. Lorieux, A. Vert, and P. Spiesser, "GeKiPe, a gesture-based interface for audiovisual performance," in *New Interfaces for Musical Expression Conference (NIME 2017)*, 2017, pp. 450–455.

[24] A. Parmentier, K. Déguernel, and C. Frei, "A modular tool for automatic soundpainting query recognition and music generation in Max/MSP," in *Sound and Music Computing Conference (SMC 2021)*, 2021.

[25] D. Ghisi, "Music across music: towards a corpus-based, interactive computer-aided composition," Ph.D. dissertation, University Paris 6, 2017.

[26] O. Ben-Tal, M. Tobias Harris, and B. L. Sturm, "How music ai is useful: Engagements with composers, performers, and audiences," *Leonardo*, pp. 1–13, 2020.

[27] T. Lubart, "How can computers be partners in the creative process: classification and commentary on the special issue," *International Journal of Human-Computer Studies*, vol. 63, no. 4-5, pp. 365–369, 2005.

[28] A. Kantosalo and A. Jordanous, "Role-based perceptions of computer participants in human-computer co-creativity," in *AISB Symposium of Computational Creativity (CC@AISB 2020)*, 2020.

[29] G. Micchi, L. Bigo, M. Giraud, R. Groult, and F. Levé, "I keep counting: An experiment in human/ai co-creative songwriting," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 263–275, 2021.

[30] C.-Z. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. J. Cai, "AI song contest: Human-AI co-creation in songwriting," in *International Society for Music Information Retrieval Conference (ISMIR 2020)*, 2020.

[31] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, *Deep learning techniques for music generation*. Springer, 2019, arXiv:1709.01620.

[32] G. Medeot, S. Cherla, K. Kosta, M. McVicar, S. Abdalla, M. Selvi, E. Rex, and K. Webster, "StructureNet: Inducing structure in generated melodies," in *International Society for Music Information Retrieval Conference (ISMIR 2018)*, 2018.

[33] Y. Zhou, W. Chu, S. Young, and X. Chen, "BandNet: A neural network-based, multi-instrument Beatles-style MIDI music composition machine," in *International Society for Music Information Retrieval Conference (ISMIR 2019)*, 2019.

[34] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," 2020.

[35] C. Allauzen, M. Crochemore, and M. Raffinot, "Factor oracle: A new structure for pattern matching," in *International conference on current trends in theory and practice of computer science*, 1999, pp. 295–310.

[36] G. Assayag and G. Bloch, "Navigating the oracle: A heuristic approach," in *International Computer Music Conference (ICMC 2007)*, 2007, pp. 405–412.

[37] J. Borg, "Somax 2: A real-time framework for human-machine improvisation," Ircam, Tech. Rep., 2019.

[38] J. Nika, K. Déguernel, A. Chemla-Romeu-Santos, E. Vincent, and G. Assayag, "DYCI2 agents: merging the "free", "reactive", and "scenario-based" music generation paradigms," in *International Computer Music Conference (ICMC 2017)*, 2017.

[39] K. Déguernel, E. Vincent, and G. Assayag, "Probabilistic factor oracles for multidimensional machine improvisation," *Computer Music Journal*, vol. 42, no. 2, 2018.

[40] D. Klakow, "Log-linear interpolation of language models," in *5th International Conference on Spoken Language Processing*, 1998, pp. 1695–1698.

[41] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to information retrieval," *ACM Transactions of Information Systems*, vol. 22, no. 2, pp. 179–214, 2004.

[42] J. Sterne and E. Razlogova, "Machine learning in context, or learning from landr, artificial intelligence and the platformization of music mastering," *Social Media + Society*, vol. 5, no. 2, 2019.

[43] B. L. Sturm, O. Ben-Tal, U. Monaghan, N. Colling, D. Herremans, E. Chew, G. Hadjeres, E. Deruty, and F. Pachet, "Machine learning research that matters for music creation: A case study," *Journal of New Music Research*, vol. 48, no. 1, pp. 36–55, 2019.

[44] J. McCormack, T. Gifford, and P. Hutchings, "Autonomy, authenticity, authorship and intention in computer generated art," in *International Conference on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART 2019)*, 2019, pp. 35–50.

MUSIC-ROBUST AUTOMATIC LYRICS TRANSCRIPTION OF POLYPHONIC MUSIC

Xiaoxue Gao
National University of Singapore, Singapore
xiaoxue.gao@u.nus.edu

Chitralekha Gupta
National University of Singapore, Singapore
chitralekha@nus.edu.sg

Haizhou Li
The Chinese University of Hong Kong, Shenzhen, China
haizhou.li@cuhk.edu.cn

ABSTRACT

Lyrics transcription of polyphonic music is challenging because singing vocals are corrupted by the background music. To improve the robustness of lyrics transcription to the background music, we propose a strategy of combining the features that emphasize the singing vocals, i.e. *music-removed* features that represent singing vocal extracted features, and the features that capture the singing vocals as well as the background music, i.e. *music-present* features. We show that these two sets of features complement each other, and their combination performs better than when they are used alone, thus improving the robustness of the acoustic model to the background music. Furthermore, language model interpolation between a general-purpose language model and an in-domain lyrics-specific language model provides further improvement in transcription results. Our experiments show that our proposed strategy outperforms the existing lyrics transcription systems for polyphonic music. Moreover, we find that our proposed music-robust features specially improve the lyrics transcription performance in metal genre of songs, where the background music is loud and dominant.

1. INTRODUCTION

While significant progress has been achieved in automatic speech recognition (ASR) [1–5] and deep learning [6, 7], lyrics transcription of polyphonic music remains unsolved. In recent years, there has been an increasing interest in lyrics recognition of polyphonic music, which has potential in many applications such as the automatic generation of karaoke lyrical content, music video subtitling, query-by-singing [8] and singing processing [9–11]. The goal of lyrics transcription of polyphonic music is to recognize the lyrics from a song that contains singing vocals mixed with background music.

The main challenge for a lyrics transcription system is that the background music interferes with the singing vocals, thereby degrading the lyrics intelligibility. Past studies have tackled the background music interference with broadly two approaches: 1) by incorporating singing vocal extraction (a *music removal* approach) [12–14] as a pre-processing module; and 2) by using the background music knowledge to enhance the model (a *music-present* approach) [15, 16].

In the *music-removal* approach, various singing vocal separation techniques have been studied to suppress the background music and extract the singing vocals from the polyphonic music for acoustic modeling [12–14, 17]. However, due to imperfection in music removal as well as the distortions associated with the inversion of a magnitude spectral representation, the extracted time-domain singing vocal signals often contain artifacts. Acoustic model trained on such extracted vocals are far from perfect [12, 16]. Another *music-removal* approach is to train acoustic model only on clean singing vocals, and use vocal extraction on test polyphonic music signals at the time of inference [18–20]. However, singing-only acoustic models need to be trained using a large amount of solo singing data [21], and the two-step procedure suffers from mismatch between the acoustic features between training and testing, thereby causing degradation of the performance of acoustic modeling in lyrics recognition.

Rather than directly applying a solo-singing acoustic model to polyphonic data, polyphonic audio adaptation [22] techniques are used to adapt a model trained on a large amount of solo singing data with a small amount of in-domain polyphonic audio files, and this in-domain adaptation is found to be outperforming the solo-singing acoustic models adapted with extracted singing vocals. This suggests the polyphonic data, i.e. singing vocals+background music, helps in learning the spectro-temporal variations of the background music more than the extracted vocals.

In the *music-present* approach, instead of removing the background music, the system is directly trained with the polyphonic music input for lyrics recognition [15, 16] . For example, Stoller et al. [15] used an end-to-end wave-U-net model to predict character probabilities directly from the polyphonic audio, while Gupta et al. [16] used a Kaldi-based acoustic model training with polyphonic music conditioned with music genre related information. These studies show that the task of lyrics acoustic modeling can benefit from the knowledge of background music.

For speech recognition [23] in noisy environment, the noise-robust speech recognition techniques include both speech enhancement techniques (that suppress the background noise) [24–26] and training with environment-matched noisy data [27–29]. Inspired by the success

Copyright: © 2022 Xiaoxue Gao et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

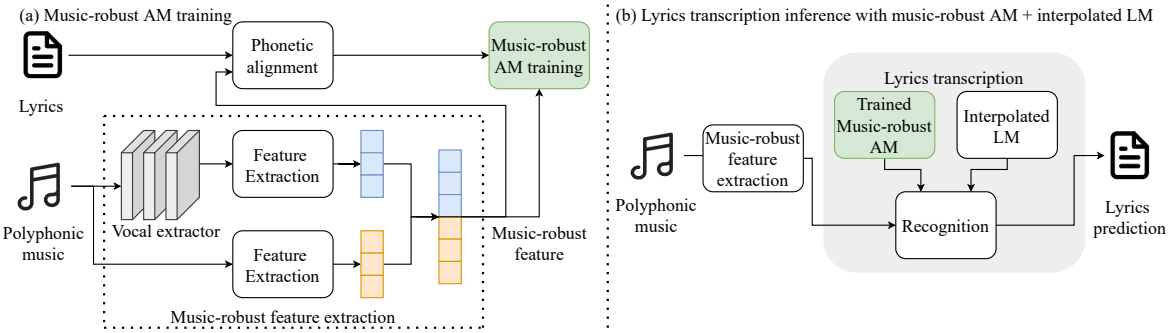


Figure 1. Model architecture of the proposed music-robust lyrics transcription framework using the music-robust AM and the interpolated LM.

of noise-robust speech recognition, we propose a combination of *music-removed* features and *music-present* features to build a *music-robust* automatic lyrics transcription (ALT) system. We believe that this combination of features will provide complementary information that will be helpful for lyrics transcription, i.e. music-removed features will focus on the lyrical content, while the music-present features will compensate for the distortions caused by the imperfect vocal extraction, and that is the focus of this paper.

Furthermore, we investigate the interpolation between a high resource speech corpus language model with an in-domain song-lyrics language model for capturing the domain-specific semantics, which yields a better performing system for the task of lyrics transcription.

The main contributions of this paper are:

- We investigate different singing vocal-related features for acoustic modeling in automatic lyrics transcription of polyphonic music;
- We propose a novel music-robust feature which is designed to enhance vocal-specific lyrical information and compensate for the corrupted temporal-spectro vocal components caused by singing vocal extractors;
- By combining the music-present features with the music-removed features, we show that lyrics intelligibility is enhanced in the presence of background music.

The rest of this paper is organized as follows. In Section 2, we will present the proposed music-robust approach. In Section 3, the experimental setting and baselines will be presented. In Section 4, we will discuss the experiments results. We will summarize the contributions of this paper in Section 5.

2. MUSIC-ROBUST AUTOMATIC LYRICS TRANSCRIPTION

We propose a music-robust automatic lyrics transcription framework for polyphonic music, which includes a music-robust acoustic model (AM) and an interpolated language model (LM).

One way of performing lyrics transcription in polyphonic music is to extract singing vocals via source separation approaches [30–33] for acoustic modeling. Features extracted from music-removed singing vocals focus on the vocals that contain lyrical information. However, singing vocals extraction systems are not perfect [30–33], and the extracted vocals suffer from the distortions caused by incomplete or distorted vocal feature prediction, as well as imperfect time-domain signal reconstruction from the estimated magnitude spectrogram. As a result, the extracted music-removed acoustic features may lose important lyrics-related information that will affect lyrics transcription performance. The polyphonic signal, on the other hand, has unaltered vocals information. However, polyphonic music often contains loud background music that overpowers the singing vocals [34]. With polyphonic features alone, the lyrics transcription performance will get affected due to the background music.

We hypothesize that a combination of polyphonic features (or music-present features) and extracted vocal features (or music-removed features) will highlight the vocal-related information while compensating for the distorted parts, thus providing a robust framework for lyrics transcription of polyphonic music. The proposed framework is illustrated in Fig. 1.

2.1 Music-robust Feature

We combine two kinds of features to form the music-robust feature as shown in Fig. 1 (a). The first feature is the *music-present* MFCCs which are obtained directly from the mixed polyphonic input audio (shown in orange in Fig. 1 (a)). The second feature is the *music-removed* MFCCs (shown in blue in Fig. 1 (a)) which are obtained from the singing-only vocals extracted from the polyphonic mixture input audio through a singing vocal extractor. The music-robust feature is formed by concatenating the music-removed MFCC feature with the music-present MFCC feature.

2.2 Music-robust Acoustic Model

We establish the inter-relations of the music-present feature and music-removed feature through the deep neural networks in a data-driven manner. These two sets of fea-

tures are stacked together as the inputs to a widely used standard ASR system that consists of an acoustic model (AM) which is trained by a factorized time-delay neural network (TDNN-F) [35].

2.3 Interpolated Language Model

Prior work have explored the language models directly trained on speech corpus text [16] or lyrics text [13, 15, 16, 20] for lyrics transcription of polyphonic. These approaches only benefit from one domain, and the cross-domain leveraging impact of linguistic contents can be explored further. In order to make better use of high resource speech text and adapt the linguistic peculiarities of lyrics of songs such as connecting words, grammar and rhythmic patterns [36] to speech text, we propose to construct an interpolated language model (LM) that bridges between a small in-domain lyrics LM trained on sung lyrics text corpus and a general LM trained on a large vocabulary speech corpus text for lyrics transcription.

We present the inference stage of music-robust automatic lyrics transcription using the proposed music-robust AM and interpolated LM in Fig. 1 (b). Given a piece of polyphonic music (background music + singing vocal), the music-robust features are extracted (Section 2.1), and with the help of the trained music-robust AM and interpolated LM, lyrics predictions are obtained by feeding the music-robust features to music-robust AM.

3. EXPERIMENTS

3.1 Datasets

As shown in Table 1, the training data for acoustic modeling consist of DALI [37] and a proprietary dataset from NUS of 517 popular English songs. DALI has 3,913 English polyphonic audio tracks [1] and comprises of 180,034 line-level audio and lyrics transcription with a total duration of 208.6 hours. The songs in the NUS proprietary dataset was split into lines automatically using the system in [16]. This dataset consists of 26,462 line-level audio and lyrics transcription with a total duration of 27.0 hours. We also used 100 songs from DALI dataset [16] which are not present in its training dataset, as a development set. We fine-tune our language model on this dev set.

We evaluate the performance of our proposed lyrics transcription framework on three widely used polyphonic test datasets – Hansen [2] [38], Mauch [39], and Jamendo [15]. The test datasets were English songs obtained from the respective authors for our research.

3.2 Acoustic Model and Singing Vocal Extraction

We employ the open-unmix system [31] for singing vocal extraction. Open-unmix is trained by parallel polyphonic mixture and clean singing vocal audio using bidirectional LSTM, and it learns to predict the magnitude spectrogram of singing vocal from the magnitude spectrogram of the corresponding mixture inputs (singing vocal+background music). It was the state-of-the-art open-source music source separation system in the source separation challenge SiSEC 2018 [40]. We use the pre-trained open-unmix model "umx" published in [31].

[1] There are a total of 5,358 audio tracks in DALI, where only 3,913 English audio links were accessible from Singapore.

[2] Total 9 songs including the song "clock" with corrected version for the errors in the ground-truth lyrics transcription.

Table 1. Dataset description.

Name	# songs	# lines	Total duration
DALI [37]	3,913	180,034	208.6 hours
NUS	517	26,462	27.0 hours
DALI-dev [37]	100	5,356	3.9 hours

The ASR system used in these experiments is trained using the Kaldi ASR toolkit [41]. The state-of-the-art Kaldi acoustic modeling pipeline consists of a context dependent phonetic alignment model, to get time-alignments between the lyrics and the audio, and an acoustic modelling network. The phonetic alignment model is GMM-HMM based trained with 40k Gaussians using MFCC features. The frame rate and length are 10 and 25 ms, respectively. For acoustic modeling, the state-of-the-art factorized time-delay neural network (TDNN-F) architecture [35] is employed.

The possible input features are as follows:

- **Poly**: AM is trained with a 40-dimensional high-resolution MFCC from polyphonic audio as input features. The phonetic alignment model input features are 39 dimensional MFCC features including the deltas and delta-deltas from polyphonic music. We refer to these poly features as *music-present* features.
- **Vocal**: AM is trained with a 40-dimensional high-resolution MFCCs from extracted vocal training set obtained by open-unmix separation [31] as input features. The phonetic alignment model input features are 39 dimensional MFCC features including the deltas and delta-deltas from extracted vocal. We refer to these features as *music-removed* features.
- **Music-robust**: AM is trained with a 40-dimensional extracted vocal high-resolution MFCCs (music-removed features) stacked with the 40-dimensional polyphonic high-resolution MFCCs (music-present features) as input features. Similarly, the phonetic alignment model input features are 39 dimensional MFCC features from Vocal model stacked with 39 dimensional MFCC features from Poly model.

The training data was augmented with speed permutation of the audio files by reducing (x0.9) and increasing (x1.1) the speed of each utterance [42]. A duration-based modified pronunciation lexicon is used that is detailed in [43]. For training the acoustic model, a frame subsampling rate is set to 3 providing an effective frame shift of 30 ms. The minibatch size is 64 where the data chunks of variable sizes of 150, 110, 100 are processed in each minibatch. Time-delay factorized neural network (TDNN-F) is used for AM

Table 2. Comparison of lyrics recognition (WER%) performance with different language models on Poly model.

	General LM	Lyrics LM	Inter LM
Hansen	62.69	51.78	**50.95**
Jamendo	61.56	57.57	**56.81**
Mauch	53.84	44.18	**43.65**

Table 3. Comparison of lyrics recognition (WER%) performance with different singing vocal-related features using the proposed interpolated LM.

	Poly	Vocal	Music-robust
Hansen	50.95	52.42	**48.33**
Jamendo	56.81	59.22	**55.73**
Mauch	43.65	45.37	**41.23**

which is trained for 3 epochs using Stochastic Gradient Descent (SGD) optimizer with initial and final learning rate of 0.00015 and 0.000015, respectively. TDNN-F [2] consists of sixteen hidden layers with 1,536 dimensions and a bottleneck dimension of 160, where each layer in the network is followed by a ReLU and batch normalization. All the acoustic models were trained according to the standard Kaldi recipe (version 5.4) [41], where the default setting of hyperparameters provided in the standard recipe was used and no hyperparameter tuning was conducted during the acoustic model training.

3.3 Language Model

The interpolated language model was built with the following steps: 1) lyrics LM was trained on the training corpus that contains the lyrics of songs; 2) general LM is trained on a large corpus of spoken English text; and 3) interpolated LM was built by interpolating between lyrics LM and general LM.

The in-domain lyrics LM is built using the lyrics corpus of the songs in training datasets (Table 1). The general LM is a 3-gram ARPA LM, pruned with threshold 3e-7 obtained from the open source of LibriSpeech language models [3] . The interpolated LM is built with the interpolation weight that yields the lowest perplexity on the DALI development set (Table 1). The interpolated LM uses the standard 3-grams with interpolated Kneser-Ney smoothing using SRILM toolkit [44].

4. RESULTS

We present our experimental results on the influence of the different input features for acoustic modeling and the different language models, and compare the proposed approach with the existing approaches as well as explore the effect of segmentation of the test datasets. Music genre analysis and error analysis under different models are also presented to better understand the results. Word confidence score analysis are presented for analysing the results. To assess the quality of lyrics transcription, we compute word error rate (WER), a standard metric of evaluation of ASR, which is the percentage of the total number of insertions, substitutions, and deletions with respect to the total number of words.

[3] http://www.openslr.org/11/

Table 4. Comparison of lyrics recognition (WER%) performance with the segmented testsets.

	Poly	Vocal	Music-robust
Hansen	50.95	52.42	48.33
Hansen-segment	45.08	45.68	**42.36**
Jamendo	56.81	59.22	55.73
Jamendo-segment	54.58	56.50	**53.21**
Mauch	43.65	45.37	41.23
Mauch-segment	40.56	43.05	**39.68**

Table 5. Lyrics transcription performance (WER%) by music genre on the test set for three models, Vocal, Poly and Music-robust.

Statistics	**metal**	**pop**	**hiphop**
# songs in Hansen	3	5	1
# songs in Jamendo	7	9	4
# songs in Mauch	8	12	0
# songs in all testset	17	27	5
Models	**metal**	**pop**	**hiphop**
Poly	60.64	37.87	**56.89**
Vocal	60.79	39.86	60.51
Music-robust	**56.60**	**36.63**	58.42

4.1 Language Model

To compare the performances of different language models, we present recognition performance results on a Poly baseline model, that is trained only on the Poly features, with different language models. As shown in Table 2, lyrics LM is significantly better than the general LM across all three test datasets. This indicates that the in-domain lyrics semantic components is beneficial to the same domain lyrics transcription task. We further found that the interpolated LM outperforms the lyrics LM in terms of word error rate (WER %) for all the test datasets in lyrics transcription, which indicates that the low-resource lyrics LM can be enhanced with the help of a large resource of general textual information. We present the rest of the experiments using the proposed interpolated LM.

4.2 Input Features

As can be seen in Table 3, system Poly outperforms the system Vocal by 2-5% showing that the music-present features that preserves the spectro-temporal variations of the vocals can help in transcription performance. Moreover, the proposed music-robust model outperforms other baseline systems across all the three test datasets, and shows 5-6 % improvement over the Poly model. This indicates that the vocal-specific features and the music-present features can complement each other and the combination of them provides more vocal-aware information that relates to the lyrical contents, therefore the combination of these two features in the music-robust acoustic model is effective in improving the accuracy of lyrics transcription in polyphonic audio.

Table 6. Analysis of lyrics recognition performance in terms of errors (insertions, deletions, and substitutions) for the segmented testsets. These are the number of words in error in each category and the percentage of each error with respect to the total number of words.

	Poly	Vocal	Music-robust
Hansen-segment	98 ins, 307 del, 790 sub	108 ins, 265 del, 838 sub	105 ins, 291 del, 727 sub
Jamendo-segment	205 ins, 962 del, 1867 sub	181 ins, 1097 del, 1863 sub	263 ins, 768 del, 1927 sub
Mauch-segment	153 ins, 636 del, 1165 sub	156 ins, 700 del, 1218 sub	147 ins, 674 del, 1091 sub
All	456 ins, 1905 del, 3822 sub	445 ins, 2062 del, 3919 sub	515 ins, 1733 del, 3745 sub
All / # total words	3.50% ins, 14.62 % del, 29.34% sub	**3.42%** ins, 15.83 % del, 30.08% sub	3.95% ins, **13.3%** del, **28.75%** sub

Table 7. Comparison of lyrics recognition example with three models. #csid means the number of correct words (C), substitutions (S), insertions (I) and deletions (D).

Model	ref GUESS THAT'S WHAT MAKES ME THE ASS I SHOULD'VE KNOWN	error pattern	#csid	WER (%)
Poly	hyp IF THERE'S ROOM MAKES ME AS I SHOULD HAVE KNOWN	S S S C C S S S S C	3 7 0 0	70
Vocal	hyp I GUESS THAT'S WHAT MAKES ME BUT AS A SHED AROUND	I C C C C C S S S S S	5 5 1 0	60
Music-robust	hyp IF THAT'S WHAT MAKES ME THAT ASS I SHOULD'VE KNOWN	S C C C C S C C C C	8 2 0 0	20

4.3 Genre Analysis

We analyse the lyrics transcription performance of different music genres in the three test sets, whose genre distribution is summarized in Table 5 according to the three broad music genre classes – pop, hiphop and metal, as given in [16]. As shown in Table 5, the music-robust system outperforms Poly and Vocal systems for metal and pop songs, and Poly model is superior to Vocal and Music-robust models for hiphop songs.

This result indicates the complementary nature of the music-present and the music-removed features through the nature of the songs. Metal songs have louder background music than other genres [34], so the music-removed features have the advantage of clearing up the background "noise", while the music-present features compensate for the distortions caused by the vocal extractor. However, hip-hop songs consist of a lot of rap, i.e. the amount of vocals is high, so the amount of distortions caused by the vocal extractor is also high (Vocal model performs the worst). Although the music-present features compensate a bit for the distortions (music robust is better than vocals only), music-robust model is worse than Poly alone. One should note here that there are only 5 hip-hop songs in the test set, so this observation on hip-hop songs may not be general enough.

4.4 Error Analysis

To better analyze the error patterns statistically, we conduct the word error analysis for different models on the segmented testsets in Table 6 and present a decoded example from different models in Table 7.

Specifically, we compare the substitution, insertion, and deletion errors in the predicted transcriptions of the Poly, Vocal, and Music-Robust systems in Table 6. In our proposed music-robust model, the substitution errors decrease for hansen and mauch datasets, and deletion errors decrease for jamendo dataset, compared to the Poly and Vocal baselines. In general, the proposed music-robust system is able to reduce deletion and substitution errors, compared with the Poly and Vocal systems. This implies that the music-robust feature is helpful to amend the distorted vocal parts that come from the vocal extraction process in Vocal system. The overall improvement from Poly system to Music-robust system indicates that the music-robust model provides better vocal-related information that is helpful to lyrics transcription compared with the Poly system. We also show an example transcription output for the same audio clip using the three models in Table 7. The output transcription of the music-robust model achieves fewer deletion and substitution errors, which further verifies our idea.

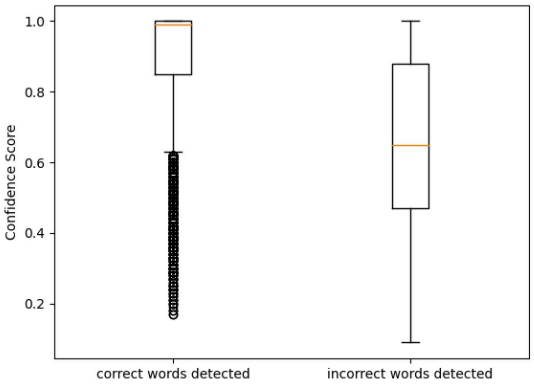


Figure 2. Word confidence score analysis over the revised segmented version of the three test sets.

4.5 Word Confidence Analysis

The confidence in prediction of a word can be a useful indicator for the purpose of assessing the quality of the predicted output, and can be used for practical applications such as song retrieval through lyrics. Kaldi provides a confidence score in prediction using minimum Bayes risk decoding [4]. The word confidence analysis is conducted on the results of the segmented test sets with the proposed

[4] https://github.com/kaldi-asr/kaldi/blob/master/egs/wsj/s5/steps/conf/get_ctm_conf.sh

Table 8. Comparison of lyrics recognition (WER%) performance with the existing approaches.

	DS [15]	CG [16]	RB1 [19]	DDA2 [45]	DDA3 [45]	GGL1 [46]	GGL2 [46]	Music-robust
Hansen-ex	-	-	84.98	77.12	66.70	50.88	49.85	**48.06**
Jamendo	77.80	59.60	86.70	72.14	73.09	60.98	62.46	**55.73**
Mauch	70.90	44.00	84.98	75.39	80.66	47.25	49.54	**41.23**

music-robust approach. As shown in Fig. 2, in general for the correctly recognized words, the confidence score is high, and the words that are incorrectly recognized show lower confidence score. This shows that when the system predicts with high confidence, the predicted words are more likely to be correct than incorrect.

4.6 Comparison with Prior Studies

We compare the proposed music-robust model with the existing approaches in Table 8. Specifically, we compare our proposed model with the systems [19, 45, 46] submitted to the lyrics transcription competition in the 16th Music Information Retrieval Evaluation eXchange International Benchmarking Competition (MIREX 2020), which is a well-known lyrics transcription challenge and the results are publicly available [5]. Since the song "clock" in Hansen datasets was cut half from the original audio for MIREX 2020 competition, we exclude the song "clock" from Hansen dataset and mark it as Hansen-ex in Table 8 for fairer comparison. [46] consider only music-present features while [19, 45] only use music-removed features for AM training, whereas our proposed music-robust system benefits from both music-present and music-removed features.

As can be found in Table 8, our proposed music-robust system shows considerable improvement for the three published test datasets in lyrics transcription performance compared to all the previous studies. We note that our model also shows 8-9% improvement over the recent top results GGL1 [46] in MIREX 2020.

We expect that transcription performance over shorter segments would be better than over the whole song, because in Viterbi decoding for long utterances, the errors tend to accumulate. Moreover, speech recognition [24–26] and lyrics transcription of monophonic music [19, 45] are generally performed on segmented data. Therefore, we segment the songs in the three test sets manually into short utterances of 20-30 seconds, which is released for the development of the research community [6]. We also further clean-up the reference lyrics with minor transcription corrections. We present the lyrics recognition performance on the revised segmented test set and compare it with the original whole song data in Table 4. We found that the revised segmented data performs consistently better than the whole song version across the baseline models and the proposed model, as expected. We will publish our segmented test sets for the research community upon paper acceptance.

[5] https://www.music-ir.org/mirex/wiki/2020:Lyrics_Transcription_Results

[6] https://github.com/xiaoxue1117/ALTP_chords_lyrics

5. CONCLUSIONS

This paper presents a novel music-robust automatic lyrics transcription approach via building music-robust features for lyrics acoustic modeling and interpolated language modeling of polyphonic music. The combination of music-present features with music-removed features offers compensation for the missed vocal components caused by singing source separation techniques, thereby conveying complementary information about the vocals while also emphasizing on the singing vocals that are highly related to lyrics intelligibility. We also investigate the interpolated LM to bridge the in-domain linguistic peculiarities with high resource out-domain information to enhance lyrics transcription performance. The experimental results show the proposed music-robust model is superior over the existing approaches on publicly available test sets.

6. ACKNOWLEDGEMENT

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project No. A18A2b0046). This work is supported by A*STAR under its RIE2020 Advanced Manufacturing and Engineering Domain (AME) Programmatic Grant (Grant No. A1687b0033, Project Title: Spiking Neural Networks). This research work is also supported by Academic Research Council, Ministry of Education (ARC, MOE), Singapore. Grant: MOE2018-T2-2-127. Title: Learning Generative and Parameterized Interactive Sequence Models with RNNs.

7. REFERENCES

[1] S. Sun, P. Guo, L. Xie, and M.-Y. Hwang, "Adversarial regularization for attention based end-to-end robust speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1826–1838, 2019.

[2] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. INTERSPEECH*, 2016, pp. 2751–2755.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[4] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Proc. IEEE ICASSP*, 2013, pp. 8614–8618.

[5] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," in *Proc. IEEE ICASSP*, 2018], pp. 5934–5938.

[6] X. Gao, X. Tian, R. K. Das, Y. Zhou, and H. Li, "Speaker-independent spectral mapping for speech-to-singing conversion," in *IEEE APSIPA ASC*, 2019, pp. 159–164.

[7] X. Gao, X. Tian, Y. Zhou, R. K. Das, and H. Li, "Personalized singing voice generation using wavernn," in *Proc. Odyssey 2020 The Speaker and Language Recognition Workshop*, 2020, pp. 252–258.

[8] A. Mesaros, "Singing voice identification and lyrics transcription for music information retrieval invited paper," in *7th Conference on Speech Technology and Human-Computer Dialogue (SpeD)*. IEEE, 2013, pp. 1–10.

[9] X. Gao, B. Sisman, R. K. Das, and K. Vijayan, "Nushlt spoken lyrics and singing (sls) corpus," in *IEEE International Conference on Orange Technologies*, 2018, pp. 1–6.

[10] B. Sharma, X. Gao, K. Vijayan, X. Tian, and H. Li, "Nhss: A speech and singing parallel database," *Speech Communication*, vol. 133, pp. 9–22, 2021.

[11] K. Vijayan, X. Gao, and H. Li, "Analysis of speech and singing signals for temporal alignment," in *in Proc. IEEE APSIPA ASC*, pp. 1893–1898.

[12] C. Gupta, B. Sharma, H. Li, and Y. Wang, "Automatic lyrics-to-audio alignment on polyphonic music using singing-adapted acoustic models," in *Proc. IEEE ICASSP*, 2019, pp. 396–400.

[13] A. Mesaros and T. Virtanen, "Automatic recognition of lyrics in singing," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2010, no. 1, p. 546047, 2010.

[14] G. B. Dzhambazov and X. Serra, "Modeling of phoneme durations for alignment between polyphonic audio and lyrics," in *Proc. SMC*, 2015, pp. 281–286.

[15] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *Proc. IEEE ICASSP*, 2019, pp. 181–185.

[16] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics transcription in polyphonic music: Does background music help?" *Proc. IEEE ICASSP*, pp. 496–500, 2020.

[17] H. Fujihara, M. Goto, J. Ogata, and H. G. Okuno, "Lyricsynchronizer: Automatic synchronization system between musical audio signals and lyrics," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1252–1261, 2011.

[18] C. Gupta, E. Yılmaz, and H. Li, "Lyrics-to-audio alignment with music-aware acoustic models," MIREX 2019.

[19] G. R. Dabike and J. Barker, "The sheffield university system for the mirex 2020: Lyrics transcription task," MIREX 2020.

[20] E. Demirel, S. Ahlbäck, and S. Dixon, "Low resource audio-to-lyrics alignment from polyphonic music recordings," *arXiv preprint arXiv:2102.09202*, 2021.

[21] E. Demirel, S. AhlbÄck, and S. Dixon, "Automatic lyrics transcription using dilated convolutional neural networks with self-attention," in *Proc. 2020 IJCNN*. IEEE, pp. 1–8.

[22] C. Gupta, E. Yılmaz, and H. Li, "Acoustic modeling for automatic lyrics-to-audio alignment," in *Proc. INTERSPEECH*, Sept. 2019.

[23] X. Feng, Y. Zhang, and J. Glass, "Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition," in *Proc. IEEE ICASSP*, 2014, pp. 1759–1763.

[24] Y.-H. Tu, I. Tashev, S. Zarar, and C.-H. Lee, "A hybrid approach to combining conventional and deep learning techniques for single-channel speech enhancement and recognition," in *IEEE ICASSP*, 2018, pp. 2531–2535.

[25] Y.-H. Tu, J. Du, and C.-H. Lee, "Dnn training based on classic gain function for single-channel speech enhancement and recognition," in *IEEE ICASSP*, 2019, pp. 910–914.

[26] K. Hermus, P. Wambacq, and H. Van Hamme, "A review of signal subspace speech enhancement and its application to noise robust speech recognition," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–15, 2006.

[27] S. Sharma, D. Ellis, S. Kajarekar, P. Jain, and H. Hermansky, "Feature extraction using non-linear transformation for robust speech recognition on the aurora database," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, vol. 2, pp. II1117–II1120.

[28] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *IEEE ICASSP*, 2013, pp. 7398–7402.

[29] Z. Zhang, J. Geiger, J. Pohjalainen, A. E.-D. Mousa, W. Jin, and B. Schuller, "Deep learning for environmentally robust speech recognition: An overview of recent developments," *ACM Transactions on Intelligent Systems and Technology*, vol. 9, no. 5, pp. 1–28, 2018.

[30] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and state-of-the art music source separation tool with pre-trained models," *Late-Breaking/Demo ISMIR*, vol. 2019, 2019.

[31] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix-a reference implementation for music source separation," *Journal of Open Source Software*, vol. 4, no. 41, p. 1667, 2019.

[32] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain," *arXiv preprint arXiv:1911.13254*, 2019.

[33] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," *arXiv preprint arXiv:1806.03185*, 2018.

[34] N. Condit-Schultz and D. Huron, "Catching the lyrics: intelligibility in twelve song genres," *Music Perception: An Interdisciplinary Journal*, vol. 32, no. 5, pp. 470–483, 2015.

[35] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks," in *Proc. INTERSPEECH*, 2018, pp. 3743–3747.

[36] J. Fang, D. Grunberg, D. T. Litman, and Y. Wang, "Discourse analysis of lyric and lyric-based classification of music." in *ISMIR*, 2017, pp. 464–471.

[37] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Dali: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm," in *ISMIR*, 2018.

[38] J. K. Hansen, "Recognition of phonemes in a-cappella recordings using temporal patterns and mel frequency cepstral coefficients," in *Proc. SMC*, 2012, pp. 494–499.

[39] M. Mauch, H. Fujihara, and M. Goto, "Lyrics-to-audio alignment and phrase-level segmentation using incomplete internet-style chord annotations," in *Proc. SMC*, 2010, pp. 9–16.

[40] SigSep, "Sisec results," https://sigsep.github.io/open-unmix/results.html#demo-tracks, 2018 (accessed Oct 4, 2021).

[41] A. Povey, D.and Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *in Proc. ASRU*, 2011.

[42] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. INTERSPEECH*, 2015, pp. 3586–3589.

[43] C. Gupta, H. Li, and Y. Wang, "Automatic pronunciation evaluation of singing," *Proc. INTERSPEECH*, pp. 1507–1511, 2018.

[44] A. Stolcke, "Srilm-an extensible language modeling toolkit," in *Seventh international conference on spoken language processing*, 2002.

[45] E. Demirel, S. Ahlback, and S. Dixon, "A recursive search method for lyrics alignment," in *IS-MIR 2020 https://www. music-ir. org/mirex/wiki/2020: MIREX2020_Results*, 2020.

[46] X. Gao, C. Gupta, and H. Li, "Lyrics transcription and lyrics-to-audio alignment with music-informed acoustic models," MIREX 2021.

SMC-22 Paper Session 5

MULTISENSORY INTEGRATION DESIGN IN MUSIC FOR COCHLEAR IMPLANT USERS

Doga Cavdir
Stanford University
cavdir@ccrma.stanford.edu

Francesco Ganis
Multisensory Experience Lab
fganis20@student.aau.dk

Razvan Paisa
Multisensory Experience Lab
rpa@create.aau.dk

Peter Williams
Multisensory Experience Lab
peter@create.aau.dk

Stefania Serafin
Multisensory Experience Lab
sts@create.aau.dk

ABSTRACT

Cochlear implant (CI) users experience several challenges when listening to music. However, their hearing abilities are greatly diverse and their musical experiences may significantly vary from each other. In this research, we investigate this diversity in CI users' musical experience, preferences, and practices. We integrate multisensory feedback into their listening experiences to support the perception of specific musical features and elements. Three installations are implemented, each exploring different sensory modalities assisting or supporting CI users' listening experience. We study these installations throughout semi-structured and exploratory workshops with participants. We report the results of our process-oriented assessment of CI users' experience with music. Because the CI community is a minority participant group in music, musical instrument design frameworks and practices vary from those of hearing cultures. We share guidelines for designing multisensory integration that derived from our studies with individual CI users and specifically aimed to enrich their experiences.

1. INTRODUCTION

While cochlear implants (CI) have achieved a high level of complexity in terms of hardware and ergonomics, training and rehabilitation programs for cochlear implant users are still lacking. This technology is quite advanced for facilitating speech perception, but music appreciation and rendering prove to be underwhelming. Specifically, most CI users report the inability to properly recognize the timbre and pitch of musical instruments, or have issues of sound localization [1, 2]. Additionally, they state to struggle with segregating the individual instruments in multi-instrument mixing [3]. In this paper, we describe a participatory design approach to designing novel technologies to help hearing impaired users' experience and appreciate music.

The hearing abilities, profiles, and perceptions significantly vary among people experiencing hearing impairments. This diversity is even wider among cochlear implant (CI) users due to "age, cognitive processing residual hearing, hearing aid use, and musical training" [4]. When approaching musical experience design for CI users, the assessment and evaluation might need to be process-oriented and individual-specific. Over the course of explorative workshops with CI users, we developed design practices and guidelines for integrating multisensory modalities to enrich their experiences with music. Our motivation derives from providing them with tools to better understand and enjoy musical features that many participant report difficulties.

Copyright: © 2022 Doga Cavdir et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. RELATED WORK

2.1 Accessible and Inclusive Design in Music

When designing accessible digital musical instruments (ADMIs) or accessible music technologies (AMTs), researchers differently approach this design and collaboration / participation process, ranging from participatory approaches to performance and improvisation.

Schroeder and Lucas discuss the process and evaluation of bespoke design approach to accessible music technologies [5]. The authors describe how bespoke designs are vital to provide access for disabled artist to music making. Lucas et al. investigate the evaluation methods for bespoke designs for music and provide their observation on assessing these designs for future ADMI designs [6]. Samuels and Schroeder study improvisation possibilities among performers of different background and abilities for increased inclusion [7] and emphasize the performance aspect in accessible and inclusive design for music.

Dickens et al. practice participatory methods to investigate real life musical interactions for people with complex disabilities and to explore potentials of embodied interactions with gesture-based technology [8]. Another participatory approach by Marti and Recupero [9] focuses on design of smart jewels beyond functionality for Deaf and Hard of Hearing (D/HoH) people with hearing aids. Similar participatory practices and their implications for rehabilitation are explored by accessibility researchers [10]; however their application to musical experience design, specifically for Deaf and Hard of Hearing participants are significantly limited. Like participatory design with D/HoH, community-engaged research with focus on music and hearing impairments is even more limited in this field. Gosine et al. discuss the importance of community building through inclusive music making and its benefits to disabled people through music theraphy [11]. They created collaboration possibilities

among persons with physical disabilities and local community musicians following a workshop format.

Frid highlights that the majority of ADMIs focus on addressing users' complex needs in terms of physical and cognitive disabilities, rather than users' experience of music who live with vision and hearing impairments [12]. By 2019, only 6% of ADMIs focused on hearing impairments, even less studied specific cases of cochlear implant use and music.

2.2 Cochlear Implant Use and Music

Cochlear implants have witnessed an impressive evolution in the last 30 years, restoring hearing to more than half a million profoundly deaf people. Their success is usually measured through speech recognition tests. Common implant systems achieve 50% - 60% accuracy after 24 month of use when tested on monosyllabic words, and close to 100% on sentences [13]. Some patients achieve spectacularly high results providing proof of what is possible with a neuroimplant in an otherwise totally deaf cochlea. Variability is high though, with standard deviations ranging from about 10% to 30%, for various studies, but results are improving, especially in patients using bilateral implants [13].

The CIs available today still have significant limitations, offering a severely impaired pitch and timbre perception. Another known limitation is the difficulty users have when presented competing sounds; CI users struggle to discriminate musical events when multiple instruments are playing, or long reverberations are present [14, 15]. Furthermore, there is a general weak representation of the fundamental frequencies (F0) for complex sounds, with difference limens ten times lower than hearing without no impairments, even when signals are below that of the CI pitch saturation limit (300Hz) [13]. As a result of these cumulative factors, the evaluation of music experience is not included as a measurement of success for the implants, as the general music experience for CI users is poor.

2.3 Multisensory Integration in Music

At the core of this project lies the principle of multisensory integration that explains how humans form coherent experiences by merging information from multiple senses [16]. For this integration to occur, the only requirement is that the stimuli are temporally overlapping; this will produce a perceptual enhancement that is strongest for the stimuli which are least effective [16].

In the specific case of auditory-tactile stimuli, recent studies demonstrate that multisensory integration can in fact occur at very early stages of cognition, resulting in supra-additive integration of touch and hearing [17–19]. This is especially useful for CI users that are shown to be better multisensory integrators []. Furthermore, research within auditory-tactile interactions has shown that tactile stimulus can influence auditory stimulus perception when presented in unison [20, 21].

Multisensory integration has been exploited extensively in previous research focusing on tactile augmentation of music; in 2009 Karam et. al. drew inspiration from previous sensory substitution vocoders and aimed to increase the audio-tactile resolution through the skin [22]. Their project resulted in a chair that provided 4 pairs of voice coil actuators arranged in an array along the back rest, following the cochlea metaphor - lower frequencies are reproduced lower than the higher ones. Each one of the actuators could reproduce one octave of the piano, from 27.5Hz to 4186 Hz [22]. They evaluated their design with respect to emotional reaction and concluded that participants enjoy the two proposed techniques more than the audio signal alone. Further upgrades to the chair resulted in a a wide spectrum of feedback, mostly positive [23].

Another chair installation was designed by Nanayakkara et al. with the help of the hearing impaired community [24]. Initially, their haptic chair had two contact speakers as haptic transducers placed under the armrest that was upgraded later with actuators directed at the lower back area and a footrest, providing a *whole body stimulation* [25]. The actuators were reproducing an amplified version of the auditory stimuli, and was always used in conjunction with sound. The chair was used successfully in long term studies (12-24 weeks) to enhance the music listening experience, as well as speech therapy for deaf children, and underlying the importance of training when users are expected to adapt a novel haptic system [25].

In 2015 a collaboration between the Deaf arts charity organization *Incloodu* and Queen Mary University resulted in an installation in the shape of an armchair and a sofa [26]. The devices used voice coil actuators placed in the backrests and armrests, and a subPac [1] under the seat. The auditory signals were spatialized from low to high areas of the backrest, and a noisy component correlated to timber was reproduced through the armrests. The structure was designed by a profoundly deaf architect, specialized in developing interiors for hard-of-hearing customers [26]. Their evaluation shows that the type of music has a great impact on the experience, with highly rhythmic music eliciting more positive reactions than music where harmonic motion was most important [26]. When music with less transients was presented, users seemed to observe the therapeutic value of vibrations. This emphasizes an important aspect of vibrotactile musical devices: they should be designed in manner that places the musical context in the spotlight.

3. RESEARCH APPROACH

The goal of this study was to (1) invite CI users into the early stages of designing novel audio-tactile displays by introducing several multisensory installations and (2) to understand the limitations of presented configurations. We performed an exploratory study, collected by a triangulation of methods: think aloud protocol, observations, and enter and exit interviews [27].

3.1 Workshop Format

Each meeting followed a predefined structure and lasted 60 - 120 minutes; for the entire duration there was one of the authors taking notes and recording the conversations. Before the meeting, the participants were requested to fill an

[1] https://subpac.com/

online survey, focusing on demographics and their past and current music listening habits. The answers from this survey formed the foundation for an semi-structured interview that was conducted before any installations were introduced; the focus was on exploring further the music engagement habits. Subsequently, the participants were guided to explore and experiment different installations described in section 4, and concluded with a shorter exit interview, summing up their feedback. Throughout the whole meeting, the participants were in contact with at least one of the authors, and were encouraged to *think aloud*.

3.2 Participants

Three participants voluntarily participated in the study, invited via open invitation on the national CI user's Facebook [2] group or via email.

Participant 1 (P1) is 52F and started losing her hearing at the age of 3, currently with no residual hearing. In 2017, she got bi-implanted with Kanso CI, experiencing a positive transition from hearing aid to cochlear implants. She likes *Fleetwood Mac*, *Dolly Parton* or *The Beatles*, but dislikes techno, classical music and heavy metal. She has background in piano and dancing (in African and Danish dances). She sings in a choir but is challenged in distinguishing and synchronizing with accompaniment, misidentifying when to start singing. She reported using a water bottle or glass in her hands to feel the vibrations in concerts.

Participant 2 (P2) is 69M with genetic hearing disability, uses a cochlear implant in his left ear, and a hearing aid in his right ear. He has experience from a musician family, in singing in a church choir, and performing competitive dancing. He likes opera, waltzes, church and classical music, and dislikes rock. More recently, he rarely listens to music. When listening to familiar music, he expresses: *"[...]my memory was another [...] I have this sort of feeling of something is in another way."*

Participant 3 (P3) is 41M. He uses a Nucleus Cochlear implant in the right ear, and near deaf in the left ear, with hearing threshold at +95dB. He has been using hearing aids since the age of 3, frequently upgrading them to higher amplification ones. When listening, he can identify when music is playing, the sex of the singer, and the instrument if the music is performed live on stage. He regularly attends to festivals, mostly for the social reasons. Lately, he enjoys listening to music for short periods of time (5 minutes) since after about 10 minutes it becomes exhausting. He mostly likes rock, especially the band *Dizzy Mizz Lizzy*.

4. DESIGN AND IMPLEMENTATIONS

4.1 Installation 1

CI users experience significant difficulties in identifying individual instruments in a musical piece [28]. In this installation, we addressed this issue by creating a multi-channel listening experience. The installation tested CI users' instrument segregation process through reproducing multi-channel recordings in a four channel speaker setup. We encourage the listeners to freely move around the room and hear individual sound sources to compare and contrast the single and multi-instrument mixings.

4.1.1 Setup

The experiment was conducted on campus at Aalborg University Copenhagen, in an anechoic room in order to prevent room reverberation altering or reducing loudspeaker directionality. The setup consisted of four *Dynaudio BM5 MKIII* loudspeakers connected to a laptop through an *Steinberg UR44C* audio interface. Each loudspeaker was fed with a dedicated output from the audio interface with only one instrument. We played multi-track recordings using Reaper - a Digital Audio Workstation (DAW) to route the instruments to independent speakers: (1) drums, (2) bass, (3) vocals, (4) keyboard or guitar alternating.

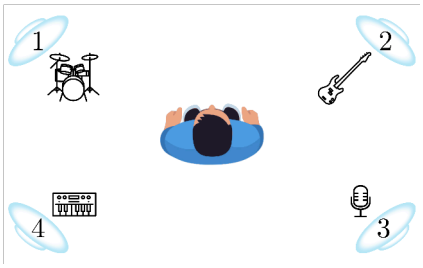


Figure 1. Scheme of installation 1.

For all the three sessions with the participants, no routing changes were applied to maintain consistency between the experiences. The dB level of each channel was set to obtain a balanced mix that allowed a hearing person to perceive all the instruments with perceived equal loudness in the center of the room by the authors. The single recordings were played without any effect such as reverberation or compression to avoid any possible confusion in the listener.

4.1.2 Experience

Once entered the room, we explained briefly what the experience was about and we let the test subject choose which music they preferred between three famous Rock, Soul and Reggae songs. Later, we proceeded setting a proper loudness level that was agreed together with the user. For all the test we set all channels to a *conversation level*.

For the first part of the experiment, we asked the subject to stand in the middle of the room and try to identify which instruments were played and from which loudspeaker they were coming from. After collecting the answers, we asked the user to walk around the room moving close to each loudspeaker to confirm or correct his/her statement about which and where instruments were played. For the second and last part, we let the subject find a sweet-spot in the room where the music sounded best for him/her. During the whole experiment the test subject was free to comment or explain at any moment their thoughts and perception of the experience.

[2] Facebook CI Group

4.2 Installation 2

A design process was undertaken to explore if and how audio-tactile feedback might be integrated into a seating installation to enhance CI users' music listening experience. We focused on providing low-frequency enhancement since CI users experience poor auditory resolution in this range.

We tested two mock-ups with 3 CI users and 3 hearing participants (including the designers). Each mock-up consisted of three components, a seat, a footrest and a hand held device, used both independently and simultaneously. All users accessed to the gain control for each actuator, through a headphone splitter used to feed the same signal to the amplifier for each transducer. Only the first user chose to manipulate the gain balance herself, while the last two provided verbal instructions to the researchers. The audio was played through either a pair of *B&W 800D* speakers for the first user, and a pair of *Mackie SRM450 + Mackie SRM1550* for the second and third participant. The users had access to the master volume knob that controlled the audio level, as well as the signal feeding the headphone amplifier (used here as a multi-channel signal splitter), thus coupling the auditory and the tactile volume.

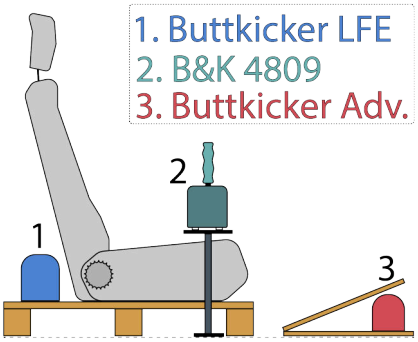


Figure 2. One configuration experienced by all participants

4.2.1 Hardware

Three types of seated installations provided different experiences: The first installation was a tactile car seat actuated by a *Buttkicker* [3] LFE that was initially powered by a *Buttkicker BKA1000* and later StageLine ST600 in bridge mode. The *BKA1000* amplifier was found to limit the higher frequencies. Both the chair and the actuator were bolted onto a wooden EUR-pallet platform, with the actuator behind the seat(see Figure 2). The actuator provided strong enough tactile feedback throughout the entire body, including the headrest. P1 and 3 hearing participants reported that it could easily felt overwhelming with higher gain.

The second and third type of seated experience shifted from a low seating position to a more upright one through a bar stool instead of the car seat, based on P1's feedback that rated the first design overwhelming. We chose the bar stool design since it affords control over the amount of weight the user applies onto, linking to the amount of feedback received. A *Buttkicker Advanced* powered by a *Buttkicker BKA300* actuated this seating. As a much smaller actuator compared to the one from the car seat, this setup required less power. The authors noticed a substantial difference between the frequency responses of the two, with a high frequency emphasis for the setup with *Buttkicker Advanced*.

P2 and P3 experienced different configurations; for P2, the actuator was bolted perpendicular to the seating area, while for P3, the actuator was fixed parallel to the ground on the side of the seating area. The side actuator configuration aimed to conduct more tactile stimuli, as P2 commented on the low intensity of the bar stool (possibly in comparison to the car seat). We observed an unexpected phenomenon in P3's setup that loud transients laterally shook the bar stool, feeling like a small "kick in the back of the chair", potentially due to the loose joints.

The footrest was designed according to H. Dreyfuss measurement recommendations and featured an inclined plane at 22° [29]. In order to have clearance for the actuator underneath the inclined plane, the footrest measured 45cm length and 60cm width. The same *Buttkicker Advanced + BKA300* combination was used, as with the bar stool. The transducer was bolted underneath the footrest, perpendicular to the ground (see Figure 2). All participants experienced the same setup.

Two handheld devices were used. A cylindrical handheld grip measuring 204mm in length and 110mm in diameter was fabricated by stacking 51 laser cut slices of 4mm HDF, following design recommendations from H. Dryeyfuss [29]. This grip was was attached to a *Brüel & Kjær(B&K) Type 4809 portable vibration exciter* (see Figure 2). The second interface, VAM (Vibrotactile Actuator for Music), was built around the Tactuator BM1C [4] [30] with an ovoid shape measuring 84mm in width, 58mm in height and 89mm depth. P1 individually tested the cylindrical grip and the VAM in combination with the seat and footrest, but the latter was deemed "not adding much" and abandoned for P2 and P3.

4.2.2 Audio Stimuli

The first audio stimulus, *Peggy Lee*'s *Fever* was presented in every Installation 2 configuration due to its clear instrument separation and the prominence of the female vocal track, matching CI users' appreciation [31]. Firstly, two different signals were sent to the handheld grips in consecutive renditions: the first identical to other actuators' signal and the second filtered to isolate the female vocal range and pitch shifted to one octave lower to skin's sensitivity range [32, 33], only applied to P1's experiment. P2 and P3 heard solo bass improvisation of *Fever* on the double bass or ukulele bass. The performance presented different playing styles (pizzicato, slapping, staccato, etc.), using the full range of the instrument, and experienced in bar stool and car seat setups. The drums accompanied to P3's experiment.

Participants selected extra audio material for their preferred setup. All three preferred the setup in Figure 2. P1 listened to *Fleetwood Mac - Dreams*, P2 *Vienna Philharmonic - An der schönen, blauen Donau* (excerpts), and P3 *Dizzy Mizz Lizzy - Silverflame*.

[3] https://thebuttkicker.com/

[4] http://tactilelabs.com/

4.3 Installation 3

We also studied participants' experience with embodied interactions using movement-based performance and in-air haptics. To simulate this experience, we discussed excerpts from a previous inclusive performance study, Felt Sound, designed for both D/HoH and hearing audience members [34]. Originally, Felt Sound, consisting a digital musical instrument, a performance setting, and a user study, was performed in-person with an 8-subwoofer speaker setup where the participants sat close to or touched the speakers. Due to time and space restrictions and COVID-19 precautions, performance excerpts [5] [6] were individually shared with the participants with two subwoofers enclosing their sitting area and facing the participant. The participants were still encouraged to interact with the speakers and feel the vibrations through touch.

We briefly described Felt Sound's motivation, concept, and performance practice. After providing the participants with its context, we presented its excerpts. Following the performance, we discussed their experience both with Felt Sound and with their own movement and music practices. Presenting a new movement-based musical concept led participants to share their own associations and experiences with movement practice and music.

5. EXPERIENCES AND RESULTS

We audio recorded the discussions with each participant and transcribed them after the study. This chapter will present a summary of these discussion sessions, focusing on their appreciation of the installations and their overall experiences.

5.1 First Participant

P1 listened using 4 vibrotactile devices: car seat, footrest, hand grip, the VAM, in 2 cases (processed and unprocessed signals) as detailed in Section 4.2.1. The audio volume was tied to the overall actuator amplitude. The researchers initially set the individual tactile amplitudes to "perceptually equal" and the listening volume to "comfortably loud", slightly over conversation level.

In the first case (listening to the processed audio), she reported how it was *"fun to feel the vibrations in the entire body"*, re-iterating her experience with the water bottle during concerts (see Section 3.2). She did not understand the mapping of the vocals to the haptic feedback, stating that she could already hear the voice through the speakers, and would not need extra stimuli representing the vocals. Additionally, she only adjusted the volume of the hand grip up several times.

When presented with the second case (listening to the unprocessed audio), she seemed more engaged in the song, grooving with the rhythm and moving to music. Similar to the first case, she experimented with slightly turning up the hand grip, footrest, and seat. When the song was over, she stated that she preferred this listening method over the first case because she can feel the melody in the footrest. She also expressed that listening to the vibrations through the chair setup could sometimes feel overwhelming.

Her perception changed over the course of the experiment. She reported that she could feel the vocals through the hand grip and the bass line (initially she assumes it was a keyboard) through the foot pad and the seat, expressing that it was fun. Although all actuators reproduced the same signal, different haptic experiences where perceived at different locations on the body, that amplified their perception of pitch and instrument type. She answered to whether she would use such a device at a concert as *"I would like to have some help from vibrations"* and explained how she sits very close to the speaker at concerts to get the haptic feedback. Furthermore, she said she would use them if *"it is trusty"*. She less emphasized her experience using the VAM compared to the other haptic listening tools, stating that it was not strong enough. We interpreted her articulations about the VAM as "not strong, relative to other actuators".

After listening to Installation 3, she discussed her experience with music and movement. This installation led her to articulate her movement practice and more embodied experiences with music such as singing. She reported that when she sings in a choir, she experiences the difficulty of identifying the onsets, specifically knowing when to start singing only by listening to the piano. She stated that she would be interested in incorporating gestures to her singing practice to assist her and to support her conductor's assistance for her. Additionally, she expressed that seeing a gesture-based performance was supporting her understanding and enjoyment of music.

5.2 Second Participant

P2 listened to *Ain't No Mountain High Enough* by *Marvin Gaye & Tammi Terrell* with Installation 1. When listening to the piece in the middle, he correctly identified the left and right channels of the instrument sources. However, he guessed the incorrect instruments at each channel. After we asked him to move closer to each speaker, he correctly identified all the instruments, including the male and female voice alternating, not being able to distinguish the lyrics. Similar to the voices, he was able to identify that the guitar and keyboards were playing together in the same channel. He was very unsure of his answers, stating that *"it's always about guessing"*. He always directed his non-implanted ear towards the speakers, making use of his hearing aid.

We lastly asked him to freely select a spot where the music sounds the best for him. He chose a spot in the middle of the 1-2 3-4 speaker pair, closer to the 1-2 speakers, and said *"... I think this must be the ideal (spot) for this kind of music that all of it is, is possible to hear."* After being exposed to all instruments individually he said that they became clearer once he separately heard and identified them. Similarly, when identifying the lyrics, he could follow them once he was told what the chorus lyrics were.

The second installation consisted of the car seat, the bar stool (with vertical actuator) the footrest and the hand grip powered by the *B&K* actuator, with the same volume settings as initially set for P1. The setup was split in two: (1) bar stool with footrest and had hand grip and (2) car seat

[5] https://tinyurl.com/2p8axhwp
[6] https://tinyurl.com/yck63zbz

with footrest and hand grip. We played the same music without any processing for the actuators. After approximately 90 seconds of listening through the first setup, we paused the listening for intermediate discussion and the participant described where he most significantly felt the vibrations: in the thigh, ankles, and up to the elbow. He provided verbose feedback regarding the locations and intensities of perceived vibrations, but limited in terms of perceptual qualities of the stimuli. He could identify the female voice and the deep bass. He also stated he could easily identify the melody.

When we asked him how the music made him feel, he said: *"It was more like a little bit sad music."* and stated how there should be more happiness in it for him to appreciate it. Furthermore, when one of the researchers played the double bass solo, P2 appreciated the live music aspect but he stated that he does not like the bass (as an instrument). He further reported that the installation was more involving but influenced by the choice of music since the music piece was not a style of music he enjoys; thus, becoming and enhancement of something he does not prefer. He requested listening to *An der schönen, blauen Donau* composed by J. Strauss. From the very first chord, the participant said *"... yeah this is much better, much better, yeah and I can feel it supports the music. So, if you like the music, this gives extra power"*. The second setup was experience only with the waltz playing, but the discussion diverted towards commercial value of musical experiences, and no feedback on the second setup was noted. He mentioned that he would not use such system (setup 2) in an concert environment, stating that *"[he] is rather conservative, and he'd prefer a regular chair, unless explicitly invited to try on in a concert hall"*.

His experience with Installation 3 varied from P1's. He less enjoyed the low frequency content of the music. He reported that he could feel the vibrations on his body but this form of listening did not enhance his experience of music. He finally stated that the gestural performance aspect of the music was effective.

5.3 Third Participant

P3 selected to listen to *Don't stop be now* by *Queen* in Installation 1. By standing in the center, he correctly identified the voice and mentioned that there was a lower volume coming from the speaker that was playing the bass line. After getting closer to each speaker, he quickly identified the voice correctly, and mislabeled the piano as guitar. When he approached the speaker playing the bass line, he experienced difficulty in identifying the instrument, asking if it was a tuba. He correctly distinguished the drums.

When we asked him to choose a favorite spot in the room he walked for several minutes, moving between speakers and overall listening area. The chosen spot spot was equally distant from speakers 1 and 2, and much further from speaker 3 and 4 that he was facing. At this spot, he stated that he could hear "a bit of everything", but only mentioning the drums, bass, and vocals. During the post-experiment discussions, we observed that he enjoyed listening to instruments separately since he could make sense of them on his own terms. He further shared his discussions with other people about the sound of bass (at concerts) that *"[he] could never distinguish [the individual instruments] because everything sounds like "mush", but it was a bit easier in this case, after hearing each instrument separately"*.

The second installation followed a similar structure to experiment with P2, only difference being the orientation of the actuator on the bar stool as described in Section 4.2.1. After about 90 seconds (before the second verse), the music was stopped and the participant rapidly mentioned that he mostly felt the hand and the bar stool did not add anything to the experience. When asked, he could not identify the valence of the song. Before resuming the music, all actuators were turned down and we slowly increased their amplitude one by one while we instructed the participant to focus on preference over actuated areas. The results were the same; he preferred the hand grip and the footrest (especially when it was turned up more). He mentioned that it's difficult to identify the mood of the song claiming that on one side it's *"slow and heavy, but the singing (voice) sounds happy"*. For the live ukulele bass performance all actuators were set to initial amplitudes; for feedback, P2 said that he preferred the lower frequencies from the footrest, but when the frequency gets higher, it's better through the hand handle. Additionally, when short and fast notes were played, he reported that it was easier to *"feel what happens"* through the hand grip. Similar to the first case, the bar stool *"did not have much to offer"* in this experience.

Moving to the second setup, the participants mentioned that *"this is much better to have it in the back, this way"* further mentioning that setup 1 felt a bit distant. During this experience, the actuators' volume was manipulated by a researcher leading to the conclusion that it's best when all 3 actuators are perceivable, and that if feels "empty", when the seat is not actuated. After the live bass performance (same as for setup 1), P3 claimed that it's fun to use the setup, but still feels like he is *"missing something"* and that he *"just misses actually being able to enjoy music"*, a fact that was not changed by using the presented setup. Nevertheless, he could *"feel"* the voice more through the hand grip, just as with setup 1. When asked whether he preferred the live performance, or the recorded one, he said that the latter one is nicer because there's more instruments, "more different sounds". This led us to and impromptu drum and bass duo performance with two of the authors, briefly jamming over the bass line from *Fever*. The participant claimed that he always thought the bass sound is coming from the drums (in live shows), but now he understands how to separate the two.

His experience with Installation 3 reflected P1's comments on the gestural performance. He reported that he never experienced a music performance where music was played by the gestures and felt on the body.

6. DISCUSSION AND FUTURE DIRECTIONS

6.1 Process-oriented Assessment on CI and Music

Due to the variance in CI users' perception, experience, and understanding of everyday sounds, speech, and music, we

believe that the experience designs should be personalized to the individual CI users and offer customization. Although CI users might share common difficulties in experiencing music such as pitch identification, source localization, and instrument segregation (auditory streaming), their priorities in addressing these challenges significantly vary from individual to individual. For example, P1 experienced hearing the nuances in pitch variances of singing however due to her music practice, she prioritize practicing onset detection and phrasing to support her singing in choir. Similarly, P2 preferred limiting his experience to the music styles he enjoys and enhancing these specific styles rather than practicing for the gaps in his music perception. Researchers and designers should consider such interpersonal differences not only in hearing profiles but also musical appreciation, engagement, and preferences. The factors such as age, hearing aid use, musical training among many others have significant influence in such design considerations when working with CI users.

Similarly, for many CI users, experiencing music is new and requires constant practice and learning. An ongoing musical engagement where users can practice where they experience difficulty in understanding music becomes crucial. Our assessment approach reflects this process of exploring and understanding CI users' hearing and engaging with technology in ways to both support their hearing development and music appreciation. Their participation in ideation and leading the design directions was crucial to the research process.

Because their reference of music is more subjective when they articulate their music perception and experience, we frequently referred to the current literature on assessing CI hearing and informed our experience design research. We believe that a more holistic approach to supporting CI users' music engagement offers more embodied approaches to listening and music-making. Developing new musical interaction experiences leads an integrated and a participatory research process rather than distinctly dividing design, assessment, and evaluation processes. Additionally, we observed that this process-oriented assessment facilitates designers to find more collaboration opportunities with CI users since finding participants in the CI community still remains one of the biggest challenges. We believe that creating a more formal organization around cochlear implant use and music can support their participation in design and research, enhancing their musical experiences.

6.2 Guidelines for Designing Multisensory AMTs

Designers who develop tactile displays for CI users can benefit from creating devices that are flexible and that can account for different musical tastes, hearing abilities, and musical engagement levels. While our sample size limits us from generalizing overall CI users' experience in the broader community, the very different requirements from each participant only underlines the need for flexibility and customization in design. Furthermore, special attention should be taken towards not creating unpleasant experiences, as it was briefly the case for P1 (tactile stimulation too powerful) and P2 (unpleasant music choice). Prior knowledge of target groups can help with the preparation, but a certain step towards this pre-study is ensuring that displays have basic controls for tactile and auditory stimuli levels and in the case of multi-actuator devices, setups have independent control for each transducer in paramount. Another helpful approach is to consider flexible or modular hardware that can be easily reconfigured according to user's needs. Through participatory action, research can explore individual requirements. Lastly, whenever possible, we suggest the integration of visual feedback in forms of gestural or movement-based performance or visualization that can support the gaps in perception from either the tactile or the auditory channel.

7. CONCLUSIONS

In this paper, we study cochlear implant (CI) users' engagement in music and ways to support their musical experiences both in listening and participating. We conduct exploratory workshops with three participants who all use cochlear implants with different hearing profiles. Based on our discussions, we addressed their individual musical needs and tested their experience in listening music through three different installation setups. Each installation investigated a different musical aspect that CI users experience difficulty perceiving. The motivation behind the installations extends beyond informing CI users about musical content but also to enrich their listening experience and musical appreciation. We discuss key findings, results, our observations on their interaction with these three listening modalities. We detail our process-oriented assessment and provide guidelines for designing multisensory integration to creating musical interaction and experiences, with specific focus on CI users. Our efforts address the lack of available resources for CI users' music perception, understanding, and enjoyment.

Music listening needs to be approached as a multifaceted experience which can be challenging and effortful for the hearing impaired individuals. Moving forward, we hope to utilize our interaction tools and listening experiences for CI users in offering them new rehabilitation and practice frameworks while supporting their musical enjoyment. We further plan to address one of the prominent research challenge and limitation we faced during our workshop series: accessing the cochlear implant users and Deaf communities. We hope to continue our work on music for hearing impairments through building communities and meaningful collaborations between CI users, musicians, designers, and researchers, as there seems to be genuine enthusiasm and interest in using hearing assistive devices s for music, from CI users.

Acknowledgments

This work is supported by NordForsk's Nordic University Hub, Nordic Sound and Music Computing Network, and the European Art Science and Technology Network.

8. REFERENCES

[1] M. F. Dorman, L. Loiselle, J. Stohl, W. A. Yost, A. Spahr, C. Brown, and S. Cook, "Interaural level differences and sound source localization for bilateral cochlear implant patients," *Ear and hearing*, vol. 35, no. 6, p. 633, 2014.

[2] M. F. Dorman, L. H. Loiselle, S. J. Cook, W. A. Yost, and R. H. Gifford, "Sound source localization by normal-hearing listeners, hearing-impaired listeners and cochlear implant listeners," *Audiology and Neurotology*, vol. 21, no. 3, pp. 127–131, 2016.

[3] J. J. Galvin III, E. Eskridge, S. Oba, and Q.-J. Fu, "Melodic contour identification training in cochlear implant users with and without a competing instrument," in *Seminars in hearing*, vol. 33, no. 04. Thieme Medical Publishers, 2012, pp. 399–409.

[4] K. Gfeller, V. Driscoll, and A. Schwalje, "Adult cochlear implant recipients' perspectives on experiences with music in everyday life: A multifaceted and dynamic phenomenon," *Frontiers in neuroscience*, p. 1229, 2019.

[5] A. Lucas, M. Ortiz, and F. Schroeder, "The longevity of bespoke, accessible music technology: a case for community," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2020, pp. 243–248.

[6] A. M. Lucas, M. Ortiz, and F. Schroeder, "Bespoke design for inclusive music: The challenges of evaluation." in *NIME*, 2019, pp. 105–109.

[7] K. Samuels and F. Schroeder, "Performance without barriers: improvising with inclusive and accessible digital musical instruments," *Contemporary Music Review*, vol. 38, no. 5, pp. 476–489, 2019.

[8] A. Dickens, C. Greenhalgh, and B. Koleva, "Facilitating accessibility in performance: participatory design for digital musical instruments," *Journal of the Audio Engineering Society*, vol. 66, no. 4, pp. 211–219, 2018.

[9] P. Marti and A. Recupero, "Is deafness a disability? designing hearing aids beyond functionality," in *Proceedings of the 2019 on Creativity and Cognition*, 2019, pp. 133–143.

[10] C. Quintero, "A review: accessible technology through participatory design," *Disability and Rehabilitation: Assistive Technology*, pp. 1–7, 2020.

[11] J. Gosine, D. Hawksley, and S. L. Quinn, "Community building through inclusive music-making," in *Voices: A World Forum for Music Therapy*, vol. 17, no. 1, 2017.

[12] E. Frid, "Accessible digital musical instruments—a review of musical interfaces in inclusive music practice," *Multimodal Technologies and Interaction*, vol. 3, no. 3, p. 57, 2019.

[13] B. S. Wilson and M. F. Dorman, "Cochlear implants: A remarkable past and a brilliant future," *Hearing Research*, vol. 242, 2008.

[14] M. D. Fletcher, N. Thini, and S. W. Perry, "Enhanced pitch discrimination for cochlear implant users with a new haptic neuroprosthetic," *Scientific Reports*, vol. 10, 2020.

[15] M. V. Certo, G. D. Kohlberg, D. A. Chari, D. M. Mancuso, and A. K. Lalwani, "Reverberation time influences musical enjoyment with cochlear implants," *Otology and Neurotology*, vol. 36, 2015.

[16] B. Stein, P. Stein, and M. Meredith, *The Merging of the Senses*, ser. A Bradford book. MIT Press, 1993. [Online]. Available: https://books.google.se/books?id=uCV9QgAACAAJ

[17] J. Foxe, G. Wylie, A. Martinez, C. Schroeder, D. Javitt, D. Guilfoyle, W. Ritter, and M. Murray, "Auditory-somatosensory multisensory processing in auditory association cortex: An fmri study," *Journal of neurophysiology*, vol. 88, pp. 540–3, 08 2002.

[18] C. Kayser, C. Petkov, M. Augath, and N. Logothetis, "Integration of touch and sound in auditory cortex," *Neuron*, vol. 48, pp. 373–84, 11 2005.

[19] *An Exploration on Whole-body and Foot-based Vibrotactile Sensitivity to Melodic Consonance*. Zenodo, Aug. 2016. [Online]. Available: https://doi.org/10.5281/zenodo.851213

[20] R. Okazaki, H. Kajimoto, and V. Hayward, "Vibrotactile stimulation can affect auditory loudness: A pilot study," 06 2012, pp. 103–108.

[21] G. Young, D. Murphy, and J. Weeter, "Haptics in music: The effects of vibrotactile stimulus in low frequency auditory difference detection tasks," *IEEE Transactions on Haptics*, vol. PP, pp. 1–1, 12 2016.

[22] M. Karam, F. Russo, and D. Fels, "Designing the model human cochlea: An ambient crossmodal audio-tactile display," *Haptics, IEEE Transactions on*, vol. 2, pp. 160–169, 07 2009.

[23] A. Baijal, J. Kim, C. Branje, F. A. Russo, and D. I. Fels, "Composing vibrotactile music: A multisensory experience with the emoti-chair," *CoRR*, vol. abs/1509.05452, 2015. [Online]. Available: http://arxiv.org/abs/1509.05452

[24] S. Nanayakkara, E. Taylor, L. Wyse, and S. Ong, "An enhanced musical experience for the deaf: Design and evaluation of a music display and a haptic chair," 04 2009, pp. 337–346.

[25] S. Nanayakkara, L. Wyse, S. Ong, and E. Taylor, "Enhancing musical experience for the hearing-impaired using visual and haptic displays," *Human-computer Interaction*, vol. 28, 01 2012.

[26] R. Jack, A. Mcpherson, and T. Stockman, "Designing tactile musical devices with and for deaf users: a case study," *Proc. of the International Conference on the Multimedia Experience of Music*, 2015.

[27] M. Someren, Y. Barnard, and J. Sandberg, *The Think Aloud Method - A Practical Guide to Modelling Cognitive Processes*. London: Academic Press, 1994.

[28] A. J. Oxenham, "Pitch perception and auditory stream segregation: implications for hearing loss and cochlear implants," *Trends in amplification*, vol. 12, no. 4, pp. 316–331, 2008.

[29] A. Tilley, H. Dreyfuss, and H. D. Associates, *The Measure of Man and Woman: Human Factors in Design*. Whitney Library of Design, 1993. [Online]. Available: https://books.google.dk/books?id=5qzHQgAACAAJ

[30] R. Paisa, J. Andersen., N. C. Nilsson, and S. Serafin, "A comparison of audio-to-tactile conversion algorithms for melody recognition," in *Baltic Nordic-Acoustic Meetings*, 2022.

[31] W. Buyens, B. V. Dijk, M. Moonen, and J. Wouters, "Music mixing preferences of cochlear implant recipients: A pilot study," *International Journal of Audiology*, vol. 53, 2014.

[32] A. Wilska, "On the vibrational sensitivity in different regions of the body surface," *Acta Physiologica Scandinavica*, vol. 31, 1954.

[33] L. A. Jones and N. B. Sarter, "Tactile displays: Guidance for their design and application," 2008.

[34] D. Cavdir and G. Wang, "Felt sound: A shared musical experience for the deaf and hard of hearing," in *Proceedings of the 20th international conference on new interfaces for musical expression (nime-20)*, 2020.

GENERATING MULTIPLE HIERARCHICAL SEGMENTATIONS OF MUSIC SEQUENCES USING ADAPTED CORRELATIVE MATRICES

Paul Lascabettes, Corentin Guichaoua, Elaine Chew
Sorbonne Université, IRCAM, CNRS, Ministère de la Culture, STMS, Paris, France
{lascabettes,guichaoua,eniale}@ircam.fr

ABSTRACT

Segmentation is an important problem for music analysis, performance, perception, and retrieval. There is often more than one way to segment a piece of music, as reflected in the multiple interpretations of a piece of music. Here, we present an algorithm that can generate multiple hierarchical segmentations of a music sequence based on approximate repeated patterns. A relation between music objects defines this approximation to generate an adapted correlative matrix (ACM). Correlative matrices are data structures for representing repeated patterns that can overlap; ACMs constrain patterns to not overlap. We propose an algorithm that extracts meaningful information from ACMs to identify segmentations in a hierarchical way. Changing the relation produces alternate hierarchical segmentations of the same sequence. The algorithm iteratively selects patterns based on their distinctiveness, i.e. if other patterns begin with the same starting note or immediately after it. We apply this method to various musical objects: a sequence of notes, chords, or bars. In each case, we define different relations on these musical objects and test the method on musical examples to produce multiple hierarchical segmentations. Given a segmentation, the relation that produces that segmentation then gives a possible explanation for that segmentation.

1. INTRODUCTION

Segmentation is an important problem for music analysis, performance, perception, and retrieval. It consists of dividing up a musical sequence into non-overlapping segments. Music segmentation has been studied in the audio and symbolic domain. However, compared to work on audio sources, there has been comparatively less work on segmentation in the symbolic domain [1]. Segmentation tasks with symbolic sources focus on the musical score (i.e. the composition). Lerdahl et al. proposed the *Generative theory of tonal music* (GTTM) [2], where they started to model segmentations in a hierarchical way. This was expanded to include aspects of harmonic tension in *Tonal Pitch Space* [3]. Separate to this, computational approaches based on the GTTM rules were developed [4]. Other algorithms, such as Chew's *Boundary Search Algorithm*, were based on tonality, using key boundaries to create segmentations [5]. More recently, the *Correlative Matrix* was used to first detect and classify patterns, then determine the best segmentation using a score function [6].

Building on these prior work, we develop an algorithm that generates hierarchical segmentations of a musical sequence. Our approach differs from traditional ones in that it provides multiple hierarchical segmentations. These hierarchical segmentations have the potential for explaining different interpretations of the same music which lead to several ways to segment a piece [7, 8]. This kind of approach thus has applications to expressive music performance, and for explaining different perceptions of the same piece.

The multiple hierarchical segmentations generate by our algorithm are based on a chosen approximation. This approximation is defined by a relation between music objects which generates the *Adapted Correlative Matrix* (ACM), a data structure introduced in this paper to represent repeated patterns without overlaps. Our algorithm iteratively selects repeated patterns based on a notion of their distinctiveness, that is to say if other repeated patterns begin at the same time than the starting note (reinforcing the beginning boundary) or immediately after it (reinforcing the end boundary). The distinctiveness criterion exploits the ACM's structure by ensuring that the selected pattern is compatible with other repeated patterns in a hierarchical way. Assuming that the beginning and the end of repeated patterns influence the segmentation of a musical sequence [9], our algorithm iteratively creates boundaries to obtain hierarchical segmentations.

We apply this method to different musical genres and music objects: a sequence of notes, chords and bars. In each case, we define several relations between music objects which yield different hierarchical segmentations. Finally, we visualise the results obtained as a tree and discuss the results. The remainder of this article is structured as follows. Section 2 generalises the existing definition of the Correlative Matrix (2.1) and introduces the Adapted Correlative Matrix (2.2) in order to work with non-overlapping repeating patterns. Section 3 describes the proposed algorithm which extracts meaningful information of the ACM to generate hierarchical segmentations. Section 4 illustrates this method with various music objects, starting with a sequence of notes (4.1), then a sequence of chords (4.2) and also with a sequence of bars (4.3). Finally, Section 5 concludes this paper.

Copyright: © 2022 Paul Lascabettes et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

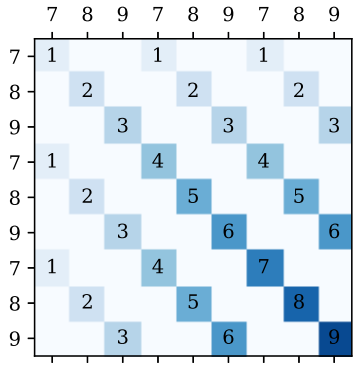


Figure 1. Correlative matrix generated by the sequence $T = (7, 8, 9, 7, 8, 9, 7, 8, 9)$.

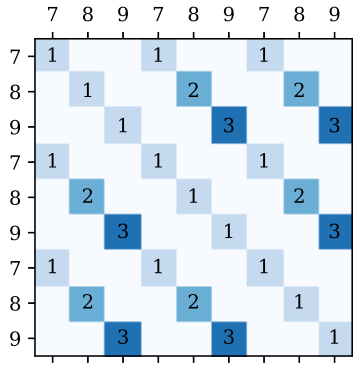


Figure 2. Adapted correlative matrix generated by the sequence $T = (7, 8, 9, 7, 8, 9, 7, 8, 9)$.

2. DEFINITION OF THE ADAPTED CORRELATIVE MATRICES

2.1 Correlative Matrix

The *Correlative Matrix* was first introduced in music processing in [10, 11] in order to detect exact repeating patterns in a sequence of pitches. For a given sequence of pitches $(p_1, \ldots, p_n)$ of length n, the first definition of the correlative matrix was an $n \times n$ matrix, where the coefficient of the i^{th} row and the j^{th} column is set to one if $p_i = p_j$. Moreover, if $p_i = p_j$ and $p_{i+1} = p_{j+1}$, meaning there is one or more repeated patterns of length two, the coefficient of the $i + 1^{th}$ row and the $j + 1^{th}$ column is set to two. By following this process, the value of each coefficient of the correlative matrix indicates the length of a repeating pattern. Compare to the self-similarity matrix used in audio-based music segmentation [12], the correlative matrix allow us to easily detect the longest repeating patterns with the maximal coefficients. Later, the correlative matrix has also been defined to allow for a sequence of intervals or a combination of pitch contours and note durations [6]. Therefore, the coefficient of the correlative matrix was modified to compare if two elements of the sequence were equal (with regard to pitch) [10, 11] or below a similarity threshold (with regard to pitch, contour and duration) [6]. This can be generalised with a symmetric and reflexive relation in order to capture a wide variety of musical objects, indeed the equality or the similarity threshold are both symmetric and reflexive relations. We then generalise the correlative matrix using the following definition:

Definition (Correlative Matrix): Let $T = (t_1, \ldots, t_n)$ be a sequence and $\equiv$ a relation on T which is reflexive ($\forall t_i \in T,\ t_i \equiv t_i$) and symmetric ($\forall t_i, t_j \in T,\ t_i \equiv t_j \Leftrightarrow t_j \equiv t_i$). The *Correlative Matrix* is an $n \times n$ matrix where the coefficient $C_{i,j}$ of the line i and the column j is defined by:

$$C_{i,j} = \begin{cases} C_{i-1,j-1} + 1, & \text{if } t_i \equiv t_j, \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

with the convention: $C_{i,j} = 0$ if i or j is negative.

For example, if $T = (7, 8, 9, 7, 8, 9, 7, 8, 9)$ and $\equiv$ is the usual equality $=$ on $\mathbb{R}$ (i.e. $7 \equiv 7$ and $7 \not\equiv 8$), the correlative matrix would be as shown in Figure 1. The following patterns are detected: $(7, 8, 9, 7, 8, 9, 7, 8, 9)$, $(7, 8, 9, 7, 8, 9)$ and $(7, 8, 9)$. The first two patterns are detected in the sequence because overlaps are allowed. In order to use the idea of the correlative matrix to identify contiguous segmentations of T, we need to adapt the definition of the correlative matrix to disallow overlaps. Granted, in music, there exist cases where the last note of a segment can be the first note of the next, but this is outside the scope of this paper.

2.2 Adapted Correlative Matrix

Here, we define the *Adapted Correlative Matrix* where nearly repeated patterns can be easily detected in a sequence T without overlaps.

Definition (Adapted Correlative Matrix): Let $T = (t_1, \ldots, t_n)$ be a sequence and $\equiv$ a relation on T which is reflexive and symmetric. The *Adapted Correlative Matrix* is an $n \times n$ matrix where the coefficient $C_{i,j}$ of the line i and the column j is defined by:

$$C_{i,j} = \begin{cases} C_{i-1,j-1} + 1, & \text{if } t_i \equiv t_j \text{ and} \\ & C_{i-1,j-1} + 1 \leq |i - j|, \\ 1, & \text{if } t_i \equiv t_j \text{ and} \\ & C_{i-1,j-1} + 1 > |i - j|, \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

with the convention: $C_{i,j} = 0$ if i or j is negative.

When $t_i \equiv t_j$, the maximal length of the pattern without overlaps is $|i - j|$, as with any longer pattern, the higher index would be in both occurrences of the pattern. Therefore, if $C_{i,j} = |i - j|$ and $t_{i+1} \equiv t_{j+1}$, instead of continuing to increment the value for $C_{i+1,j+1}$ and detecting overlapping patterns, in the adapted correlative matrix we restart at $C_{i+1,j+1} = 1$.

The adapted correlative matrix for the sequence $T = (7, 8, 9, 7, 8, 9, 7, 8, 9)$ and $\equiv$, the equality on T, is presented in Figure 2. The longest detected pattern is now:

$(7, 8, 9)$. Unlike the case with the correlative matrix, the pattern $(7, 8, 9, 7, 8, 9)$ is not detected because this pattern is repeated with overlapping.

The correlative matrix was successfully used to detect nearly repeated patterns in a musical sequence, allowing for overlaps. However, by defining the adapted correlative matrix we can retain the fundamental idea of the correlative matrix while avoiding overlaps, which is required for the music segmentation task.

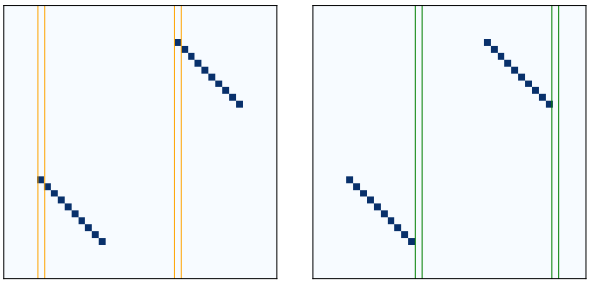
3. ALGORITHM FOR EXTRACTING HIERARCHICAL SEGMENTATIONS USING THE ADAPTED CORRELATIVE MATRIX

Let ACM be the Adapted Correlative Matrix of a sequence $T = (t_1, \ldots, t_n)$ with $\equiv$, a reflexive and symmetric relation. We describe here an algorithm that will extract data from the ACM in order to identify segmentations of T in a hierarchical way.

- **Step 1: Select the longest repeating patterns**
  The *longest repeating patterns* are defined by all the pairs $(t_{i-C_{i,j}}, \ldots, t_i)$ and $(t_{j-C_{i,j}}, \ldots, t_j)$, where $C_{i,j} = \max(\text{ACM})$ (the value of the maximal coefficients of the ACM). Often there will be more than one pattern tied for longest.

- **Step 2: Select the most distinct pair**
  Among all the detected pairs from step 1, the *most distinct one* is the one that maximises the number of repeating patterns that begin at the same time than the starting note (reinforcing the beginning boundary) or immediately after the two patterns of the pair (reinforcing the ending boundary). That is to say, we choose the pair that maximises the number of coefficients equal to 1 in the columns $i - C_{i,j}$ and $j - C_{i,j}$ as illustrated in Figure 3(a) (patterns that start at the same times as the pair) and in the columns $i + 1$ and $j + 1$ illustrated in Figure 3(b) (patterns that start just after the pair).

- **Step 3: Remove the most distinct pairs from the ACM**
  We then update the ACM by removing the most distinct pair and add boundaries to the segmentation at the beginning and end of the most distinct patterns. All coefficients of the most distinct pair become equal to 0 except for the first coefficient which remains equal to 1 (this will be useful in step 2 for future iterations). Moreover, in order to have hierarchical segmentations, if a coefficient $C_{i',j'} > 1$ is on the same column or line as the beginning or the end of a pattern of the most distinct pair, this coefficient will be equal to 1 and $C_{i'+k,j'+k}$ will be $k + 1$ while $C_{i'+k,j'+k} > C_{i'+k-1,j'+k-1}$, which is illustrated in Figure 4(d). With this, a pattern that contains a boundary will be divided in two. This creates a boundary that will remain for the next segmentations and we will thus obtain hierarchical segmentations. Finally, we go back to step 1 until there is no coefficient greater than 1 in the ACM.



(a) Detection of patterns that start at the same time as the pair.

(b) Detection of patterns that start just after the end of the pair.

Figure 3. Criteria for step 2 of the algorithm to choose the most distinct pair of patterns among the longest ones.

Let us take an example with the sequence of real numbers

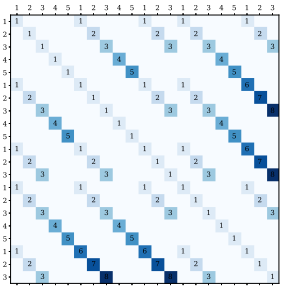
$$T = (1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 1, 2, 3, 4, 5, 1, 2, 3), \quad (3)$$

with $\equiv$ as the equality on T. The first loop of the algorithm is illustrated in Figure 4. The ACM is represented in Figure 4(a). The maximal coefficient is 8, and two pairs of longest repeating patterns are detected. The pair that maximises the number of patterns that start at the same time and right after the end of the pair is shown in Figure 4(c). Finally, this pair is removed and boundaries are created in Figure 4(d).

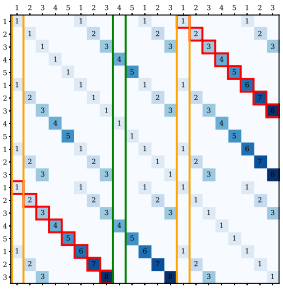
A tree visualisation can be computed to visualize hierarchical segmentations. By representing the length of the detected patterns (the most distinct pairs), the tree visualisation of the previous sequence T (with $\equiv$ the equality) is shown in Figure 5. In this case, our algorithm detects five segmentations that are hierarchically structured. The first segmentation is the sequence itself, so the length of T (here 21) is represented at the top of the tree. The last segmentation is: $(t_1)(t_2)\ldots(t_n)$ where each (t_i) is of length 1, represented by the bottom line $1/1/\ldots/1$ of the tree. The three other detected segmentations of T are:

- (1,2,3,4,5),(1,2,3,4,5,1,2,3),(1,2,3,4,5,1,2,3) represented by the line 5/8/8;
- (1,2,3,4,5),(1,2,3,4,5),(1,2,3),(1,2,3,4,5),(1,2,3) represented by the line 5/5/3/5/3; and,
- (1,2,3),(4,5),(1,2,3),(4,5),(1,2,3),(1,2,3),(4,5),(1,2,3) represented by the line 3/2/3/2/3/3/2/3.

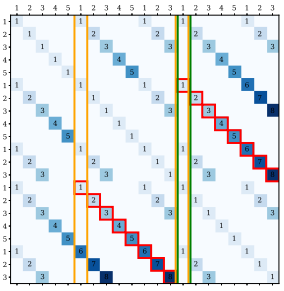
Hierarchical segmentations of musical notes has been studied by Lerdahl and Jackendoff based on different rules on the proximity between notes, repetitions and strong/weak accent of the rhythm in the GTTM [2]. They also represented hierarchical segmentations using a tree visualization. However, the method developed in this paper handles music objects other than notes and is able to propose multiple hierarchical segmentations based on the chosen relation between these objects.



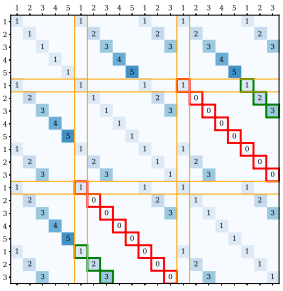
(a) **Step 1:** Two pairs of patterns of length 8 are detected in the ACM.



(b) **Step 2:** For the first pair, 10 patterns start at the same time and 1 pattern starts after the end of the pair (total: 10+1=11).



(c) **Step 2:** For the second pair, 9 patterns start at the same time and 5 patterns start after the end of the pair (total: 9+5=14). This is the most distinct pair.



(d) **Step 3:** The most distinct pair (in red) is removed and boundaries are created at the orange columns and lines (changes in green).

Figure 4. First loop of the algorithm illustrated, these three steps are executed until the coefficients of the ACM are only 0 or 1.

4. APPLICATIONS TO SYMBOLIC MUSIC REPRESENTATIONS

The algorithm presented in Section 3 generates hierarchical segmentations of a sequence with a relation which characterises the chosen approximation for detecting when two patterns are nearly the same. In this section, we apply this algorithm to various symbolic music representations: a sequence of notes (Section 4.1), a sequence of chords (Section 4.2) and a sequence of bars (Section 4.3). In each case, we propose several relations which lead to multiple hierarchical segmentations and we illustrate the results using tree visualisations.

4.1 When T Is a Sequence of Notes

Let T be a sequence of musical notes. There are many ways to define a musical note. For example, a note can be defined as a triplet (p, o, d), as in [13], where p is the pitch, o the onset and d the duration of the note. This can also be enriched to five parameters (o, p, mp, d, v) [14], where mp is the *morphetic pitch* [15] and v the voice where the note occurs. It is also possible to add other parameters such as the velocity, *general pitch interval representation* [16], etc. Among all the different parameters, we first choose here to define a note by its *interval* $\Delta p_i = p_{i+1} - p_i$ (p_i being the pitch of the i^{th} note), then we have $T = (\Delta p_1, \ldots, \Delta p_n)$.

For example, let T be the sequence of intervals of the first 32 intervals of the Prelude in C major from the *Well-tempered Clavier* by Johann Sebastian Bach (BWV 846). The score is represented in Figure 6. In this case, $T = (4, 3, 5, ..., 5, 3, -18)$. A key advantage of this representation is that it is invariant to transpositions.

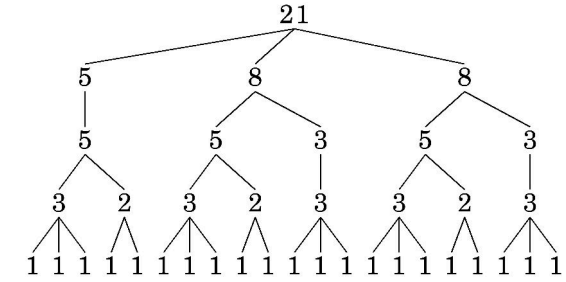


Figure 5. Tree visualisation of the five segmentations in a hierarchical way of the sequence from Equation 3.

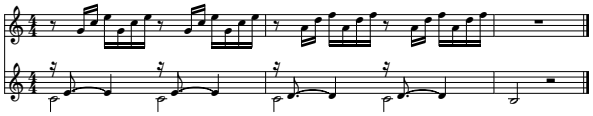



Figure 6. First 32 intervals of J. S. Bach's Prelude in C major, BWV846.

There are also different choices for the relation $\equiv$, because they are many ways to define similarity between intervals [17]. Here we will use a strict interval equality [18], a similarity threshold [6] and an "up/down" relation which defines the melodic contour [19].

Let Δp_i and Δp_j be two intervals of T, we can then define several symmetric and reflexive relations $\equiv$ for T.

- **Strict Intervals Relation:**

$$\Delta p_i \equiv_{st} \Delta p_j \Leftrightarrow \Delta p_i = \Delta p_j. \quad (4)$$

- **Similarity Threshold Intervals Relation:**

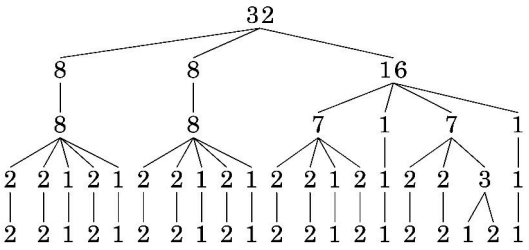
$$\Delta p_i \equiv_{sim} \Delta p_j \Leftrightarrow |\Delta p_i - \Delta p_j| \leq \lambda. \quad (5)$$

- **Melodic Contour Relation:**

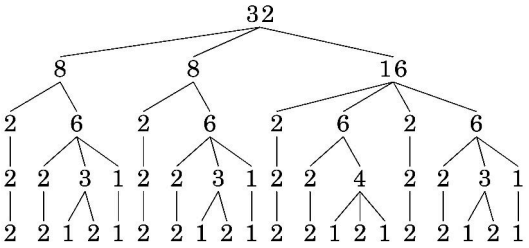
$$\Delta p_i \equiv_{mc} \Delta p_j \Leftrightarrow \mathrm{sgn}(\Delta p_i) = \mathrm{sgn}(\Delta p_j), \quad (6)$$

where sgn is the sign function defined by $\mathrm{sgn}(x) = 1, -1$ or 0 if $x > 0, x < 0$ or $x = 0$.

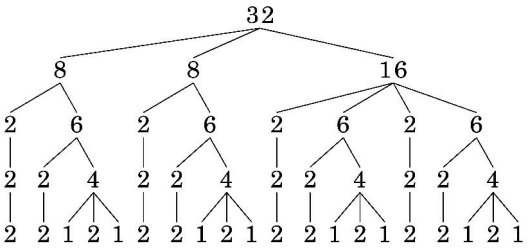
The results of our algorithm for these different relations are illustrated in Figure 7 (the strict interval relation, the similarity threshold intervals relation for $\lambda = 1, 3, 7$ and the melodic contour relation). As before, the different hierarchical segmentations are represented as trees. The ACMs are also represented in Figure 8 for these different relations. It is interesting to note that the less strict the relation is, the more "regular" the tree becomes, until we get the melodic contour relation with very regular and symmetrical segmentations.



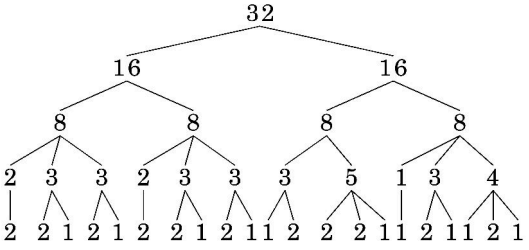
(a) Tree based on the Strict Intervals Relation.



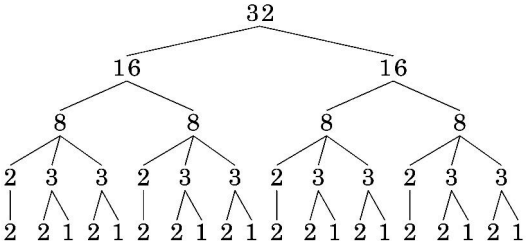
(b) Tree based on the Similarity Threshold Intervals Relation for $\lambda = 1$.



(c) Tree based on the Similarity Threshold Intervals Relation for $\lambda = 3$.

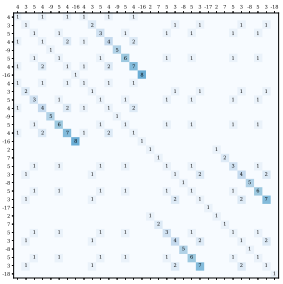


(d) Tree based on the Similarity Threshold Intervals Relation for $\lambda = 7$.

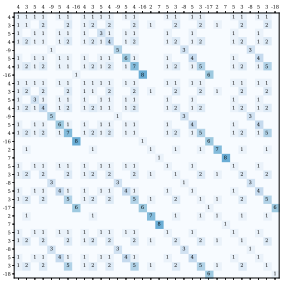


(e) Tree based on the Melodic Contour Relation.

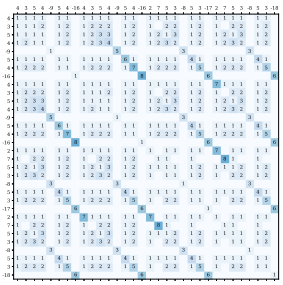
Figure 7. Different trees of the same sequence of notes T (BWV 846) when the relation between intervals changes.



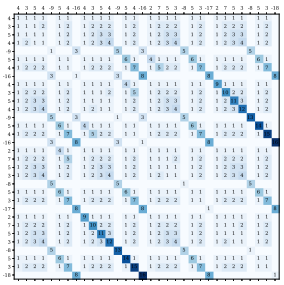
(a) ACM based on the Strict Intervals Relation.



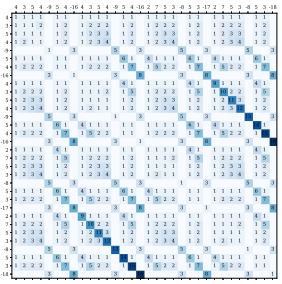
(b) ACM based on the Similarity Threshold Intervals Relation for $\lambda = 1$.



(c) ACM based on the Similarity Threshold Intervals Relation for $\lambda = 3$.



(d) ACM based on the Similarity Threshold Intervals Relation for $\lambda = 7$.



(e) ACM based on the Melodic Contour Relation.

Figure 8. Different ACMs of the same sequence of notes T (BWV 846) when the relation between intervals changes.

In order to work with the durations of the notes, we also applied the algorithm to a sequence of notes defined by $x = (\Delta p, l)$ where l is the length of the note. We defined the relation between two notes $x_i = (\Delta p_i, l_i)$ and $x_j = (\Delta p_j, l_j)$ by:

$$x_i \equiv x_j \Leftrightarrow |\Delta p_i - \Delta p_j| \leq 3 \text{ and } l_i = l_j. \qquad (7)$$

The results of the algorithm is illustrated in Figure 9 with the introduction of Chopin's Mazurka Op.7 No.1. The different colors represent different segmentations. These nearly repeated patterns could also be interesting for performance. For example, the identified patterns could be ones that might be highlighted through prosodic variations like dynamic accents; because the patterns are repeated and distinctive, they could also sound perceptually plausible.

4.2 When T Is a Sequence of Chords

Let $T = (c_1, \ldots, c_n)$ be a sequence of chords where chords of root C are labelled from the set $\hat{\mathrm{C}} = \{\mathrm{C}, \mathrm{C}^6, \mathrm{C}^7, \mathrm{C}^7_M,$

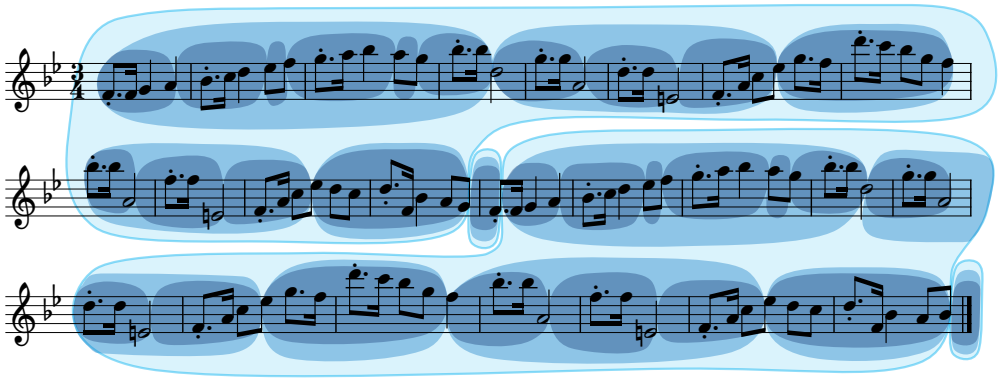


Figure 9. Detected hierarchical segmentations of the melody in Chopin's Mazurka Op.7 No.1 as marked in the score.

C_m, C_m^6, C_m^7, C_m^{M7},C+,C+^{7},C^o,C^{o7},C$^\emptyset$} and similarly for other roots. We could define the relation between two chords by strict equality ($c_i \equiv_{st} c_j \Leftrightarrow c_i = c_j$) to have a strong constraint between chords, but this runs the risk of being overly sensitive to small ornamental changes.

Let us instead define the ChordType function of a chord $c_i \in \hat{C}$ (and similarly for other roots) by:

$$\text{ChordType}(c_i) = \begin{cases} C_{maj} & \text{if } c_i = C, C^6, C^7, C_M^7, \\ C_{min} & \text{if } c_i = C_m, C_m^6, C_m^7, C_m^{M7}, \\ C_{aug} & \text{if } c_i = C+, C+^7, \text{ and} \\ C_{dim} & \text{if } c_i = C^o, C^{o7}, C^\emptyset. \end{cases} \tag{8}$$

For example $\text{ChordType}(C_m^7) = C_{min}$ or $\text{ChordType}(F\sharp^7) = F\sharp_{maj}$. Let c_i and c_j be two chords of T, we can then define the ChordType Relation $\equiv_{ct}$ on T by:

$$c_i \equiv_{ct} c_j \Leftrightarrow \text{ChordType}(c_i) = \text{ChordType}(c_j). \tag{9}$$

Some other relations, which we will not develop here, could include threshold relations based on the distance of chords within the Tonnetz [20] or one of the parsimonious relations defined by Douthett and Steinbach [21].

Let us take as example the song *In My Life* from The Beatles, released in 1965. All the songs from The Beatles are annotated at `http://isophonics.net` with: structural segmentation, key changes, chords, and beats. According to this database, the chords of the song are: $T =$ (A, E, A, E, A, E, F$_m\sharp$, A^7, D, D$_m$, A, A, E, F$_m\sharp$, A^7, D, D$_m$, A, F$_m\sharp$, D, G, A, F$_m\sharp$, B, D$_m$, A, A, E, A, E, F$_m\sharp$, A^7, D, D$_m$, A, A, E, F$_m\sharp$, A^7, D, D$_m$, A, F$_m\sharp$, D, G, A, F$_m\sharp$, B, D$_m$, A, A, E, F$_m\sharp$, A^7, D, D$_m$, A, A, E, F$_m\sharp$, A^7, D, D$_m$, A, F$_m\sharp$, D, G, A, F$_m\sharp$, B, D$_m$, A, A, E, D$_m$, A, E, A).

The results of the algorithm applied to this sequence T with the relation $\equiv_{ct}$ defined in (9) are represented in Figure 10. We can compare the hierarchical segmentations obtained by our algorithm with the annotated structure from the database represented in Figure 11.

We can see that the second line (2/2/22/2/22/22/6) contains three main sections of length 22. Each of these sections corresponds to `Verse/Bridge` and the remaining sections of length 2 and 6 map to the `Intro`, `Half-intro` and `Outro`. In the third line, i.e. (2/2/7/7/8/2/7/7/8/7/7/8/6), the `Verse` section is subdivided into two sequences of 7 chords and the `Bridge` is a sequence of 8 chords.

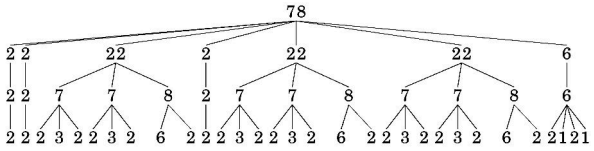


Figure 10. Hierarchical segmentations of the Beatles' *In My Life* based on the chord sequence from `http://isophonics.net`.

```
0.000   0.416   :   silence
0.416   9.616   :   intro
9.616   28.302  :   verse
28.302  46.719  :   bridge
46.719  51.438  :   half-intro
51.438  70.206  :   verse
70.206  88.700  :   bridge
88.700  107.253 :   verse_(instrumental)
107.253 125.659 :   bridge
125.659 143.715 :   outro
143.715 147.973 :   silence
```

Figure 11. Annotated segmentation of *In My Life* from The Beatles.

4.3 When T Is a Sequence of Bars

Let $T = (b_1, \ldots, b_n)$ be a sequence of musical bars. Each bar contains a set of notes (the number of notes can be different from one bar to the next). One way to define a relation $\equiv$ between two bars b_i and b_j is to consider the number of common notes between b_i and b_j. For example, b_i and b_j can be considered in relation if they share at least 50% of their notes, that is to say:

$$b_i \equiv b_j \Leftrightarrow \frac{2|b_i \cap b_j|}{|b_i| + |b_j|} \geq 0.50, \tag{10}$$

where $|b_i|$ is equal to the number of notes of the bar b_i.

With this definition, we compute the hierarchical segmentations of the song as in the previous section i.e.: *In My Life* from the Beatles. The results are represented in Figure 12. The results are similar to Figure 10 with three main sections of equal length for the second line which correspond to the `Verse/Bridge`. The `Intro`, `Half-intro` and `Outro` are also represented on this line. Note that some bars contain more than one chord, e.g. in the `Verse` and `Outro`, which is why there are more chords than bars between Figure 10 and Figure 12.

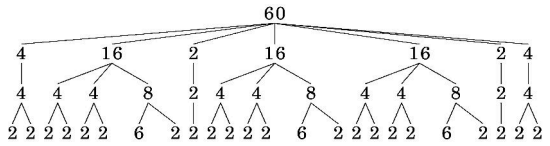


Figure 12. Hierarchical segmentations of *In My Life* from The Beatles based on a sequence of bars.

It is possible to change the relation $\equiv$ between two bars b_i and b_j by adapting the 50% threshold. Also, we can determine the musical chord or key of a bar using a key-finding function, Key, which can be based on the *Krumhansl-Schmuckler Key-Finding Algorithm* [22] or the *Spiral Array Model* [23]. Then, we can define the Key (similar for Chord) Relation $\equiv_{key}$ between two bars b_i and b_j by:

$$b_i \equiv_{key} b_j \;\Leftrightarrow\; \mathrm{Key}(b_i) = \mathrm{Key}(b_j). \tag{11}$$

In this section, we have demonstrated the usefulness of the proposed method with various music objects from symbolic music representations. Several such relations that are musically pertinent have also been presented, but many more can be conceived, including relations that would apply to audio-based objects, for instance, between two frames of a spectrogram. By adjusting which relation to focus on, it is possible to access a much broader meaning of what constitutes a repeated pattern, allowing this approach to be applied to music genres which do not typically exhibit the strong repetitions that are usually required, while preserving the algorithm's lightweight advantage.

5. CONCLUSION

In this paper, we have further expanded the generalisation of *Correlative Matrices* [11] that began with their parametric extension [6] to accept a wide family of relations that define the degree of nearness required to be considered a repeat of a pattern. In order to avoid overlapping patterns, we have introduced the *Adapted Correlative Matrix*, a data structure which represents the repeated patterns of a musical sequence without overlaps. We then defined the novel *distinctiveness* criterion, which characterises the number of repeated patterns that starts at the same time or at the end of a pattern, and proposed an algorithm that iteratively selects patterns based on their distinctiveness to generate hierarchical segmentations.

The representation and algorithms proposed are highly efficient and can scale readily to very large datasets. The examples have shown the versatility of this theory. However, at the moment, human knowledge or feedback are still very important in order to pick a representation and relation that is appropriate for the piece and the desired outcome. We expect that this representation would be a powerful tool for machine learning methods that can tailor relation operators to specific music input and find explanatory models for music segmentation.

Acknowledgments

Paul Lascabettes is funded by a Contrats Doctoraux Spécifiques pour Normalien-nes (CDSN) scholarship. This work is part of the COSMOS project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 788960).

6. REFERENCES

[1] M. Giraud, R. Groult, and F. Levé, "Computational analysis of musical form," in *Computational Music Analysis*. Springer, 2016, pp. 113–136.

[2] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music, reissue, with a new preface*. MIT press, 1996.

[3] F. Lerdahl, "Tonal pitch space," *Music perception*, pp. 315–349, 1988.

[4] D. Temperley, *The cognition of basic musical structures*. MIT press, 2004.

[5] E. Chew, "The spiral array: An algorithm for determining key boundaries," in *International Conference on Music and Artificial Intelligence*. Springer, 2002, pp. 18–31.

[6] B. Rafael and S. M. Oertl, "Mtssm–a framework for multi-track segmentation of symbolic music," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 4, no. 1, pp. 7–13, 2010.

[7] N. Cook, "Performance analysis and chopin's mazurkas," *Musicae scientiae*, vol. 11, no. 2, pp. 183–207, 2007.

[8] P. Lascabettes, C. Agon, M. Andreatta, , and I. Bloch, "Computational Analysis of Musical Structures based on Morphological Filters," in *International Conference on Mathematics and Computation in Music*, Altanta, GA, USA, 2022.

[9] E. Cambouropoulos, "Musical parallelism and melodic segmentation:: A computational approach," *Music Perception*, vol. 23, no. 3, pp. 249–268, 2006.

[10] J.-L. Hsu, A. L. Chen, and C.-C. Liu, "Efficient repeating pattern finding in music databases," in *Proceedings of the seventh international conference on Information and knowledge management*, 1998, pp. 281–288.

[11] J.-L. Hsu, C.-C. Liu, and A. L. Chen, "Discovering nontrivial repeating patterns in music data," *IEEE Transactions on multimedia*, vol. 3, no. 3, pp. 311–325, 2001.

[12] J. Foote, "Visualizing music and audio using self-similarity," in *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, 1999, pp. 77–80.

[13] M. Giraud, R. Groult, and F. Levé, "Subject and counter-subject detection for analysis of the well-tempered clavier fugues," in *International Symposium on Computer Music Modeling and Retrieval*. Springer, 2012, pp. 422–438.

[14] D. Meredith, K. Lemström, and G. A. Wiggins, "Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music," *Journal of New Music Research*, vol. 31, no. 4, pp. 321–345, 2002.

[15] D. Meredith, "The computational representation of octave equivalence in the western staff notation system," in *In Cambridge Music Processing Colloquium*. Citeseer, 1999.

[16] E. Cambouropoulos, "A general pitch interval representation: Theory and applications," *Journal of New Music Research*, vol. 25, no. 3, pp. 231–251, 1996.

[17] M. Giraud, R. Groult, E. Leguy, and F. Levé, "Computational fugue analysis," *Computer Music Journal*, vol. 39, no. 2, pp. 77–96, 2015.

[18] L. Smith and R. Medina, "Discovering themes by exact pattern matching," in *Proceedings of the International Conference on Music Information Retrieval*. Citeseer, 2001.

[19] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by humming: Musical information retrieval in an audio database," in *Proceedings of the third ACM international conference on Multimedia*, 1995, pp. 231–236.

[20] C. L. Krumhansl, "Perceived triad distance: Evidence supporting the psychological reality of neo-riemannian transformations," *Journal of Music Theory*, vol. 42, no. 2, pp. 265–281, 1998.

[21] J. Douthett and P. Steinbach, "Parsimonious graphs: A study in parsimony, contextual transformations, and modes of limited transposition," *Journal of Music Theory*, pp. 241–263, 1998.

[22] D. Temperley, "What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered," *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.

[23] E. Chew, "Modeling tonality: Applications to music cognition," in *Proceedings of the 23rd Annual Meeting of the Cognitive Science Society*, 2001, pp. 206–211.

Deep HRTF Encoding & Interpolation : Exploring Spatial Correlations using Convolutional Neural Networks

Devansh Zurale
UC San Diego
dzurale@ucsd.edu

Shahrokh Yadegari
UC San Diego
sdy@ucsd.edu

Shlomo Dubnov
UC San Diego
sdubnov@ucsd.edu

ABSTRACT

With the advancement in Deep Learning technologies, computers today are able to achieve unimaginable success in several domains involving images and audio. One such area in 3D audio where the applications of deep learning can be promising is in binaural sound localization for headphones, which requires individualized and accurate representations of the filtering effects of the anthropometric measurements of a listening body. Such filters often are stored as a set of Head Related Impulse Responses (HRIRs) or in their frequency domain representations, Head Related Transfer Functions (HRTFs), for specific individuals. A challenge in applying deep learning networks in this area is the lack of availability of vast numbers of complete and accurate HRTF datasets, which is known to cause networks to easily over-fit to the training data. As opposed to images, where the correlations between pixels are more statistical, the correlations that HRTFs share in space are expected to be more a function of the body and pinna reflections. We hypothesize that these spatial correlations between the elements of an HRTF set could be learned using Deep Convolutional Neural Networks (DCNNs). In this work, we first present a CNN-based auto-encoding strategy for HRTF encoding and then we use the learned auto-encoder to provide an alternate solution for the interpolation of HRTFs from a sparse distribution of HRTFs in space. We thereby conclude that DCNNs are capable of achieving results that are comparable to other non deep learning based approaches, in spite of using only a few tens of data points.

1. INTRODUCTION

With the increasing demand for Virtual Reality (VR) technology, there is an increasing need to accurately model sound localization. When a sound source originates from a particular point in space around the head, it undergoes different sets of reflections off of body parts and the pinna before reaching each ear. This difference in reflections, along with the difference in times of arrival and the amplitudes of arrival of the sound source at each ear is what is understood to give humans the perception of direction of the sound source. In order to then add direction to a sound source being played through headphones, one needs to filter the sound source with the transfer functions between the desired points of origination of the sound source and the points at which the sound signal enters each ear. These transfer functions, known as the Head Related Transfer Functions (HRTFs), or as their time domain representations, the Head Related Impulse Responses (HRIRs), are unique for every point in space, for each ear and for every individual.

To allow for the sound source to be perceived as originating from every direction in space, every individual would need to measure and store the HRTFs for each ear, ideally for every point in space around the head, or practically at least sampled at several hundred points. However, having to measure as many HRTFs for an individual is a time consuming and tedious process that requires very sophisticated and expensive equipment, making it impractical for consumers to record their own sets of HRTFs. One way then to be able to give binaural spatial audio experience to the listeners is to either provide a generic non-individual HRTF, or present the users with several non-individual HRTFs to pick from, the one that works best. Such non-individual HRTFs however, are known to introduce noticeable localization errors [1] which motivates the research in HRTF individualization (also referred to as personalization in literature), in order to estimate the individual HRTFs of a subject without having to measure the HRTFs for every individual.

A recent survey [2] summarizes the various methods that have been proposed towards the task of HRTF personalization. Some early works such as [3] aimed at selecting the best fitting HRTF-set from a larger database of HRTFs with anthropometric data as input features. Such approaches would perceptually work only if the user's anthropometric features are close enough to atleast one of the subjects in the database. Hence, some other works later such as [4] used linear regression techniques which provided better results than HRTF selection. Going further, not constraining to linearity, in recent years, several deep learning techniques have been proposed for the individualization of HRTFs. A recent paper [5] provides a great summary of the various deep learning methods that have been devised in this field. Some works such as [6–8] proposed using fully connected networks to predict HRTFs given an input of the anthropometric features, while some other works such as [9, 10] have proposed using perceptual listening feedback to train the network.

Copyright: © 2022 Devansh Zurale et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

1.1 Problem Statement A - HRTF Encoding

Using deep learning for predicting HRTFs however, comes with a challenge that the number of available and complete HRTF datasets is limited. To avoid over-fitting then in such situations, it is advantageous to reduce the number of parameters being used in the neural network. This is often achieved by reducing the number of parameters to predict, by encoding the HRTFs into a sub-dimensional space. One approach for dimensionality reduction was proposed by [11], through using Principle Component Analysis (PCA) to project the HRTFs linearly onto dimensions with maximum variance. A suggestion was later reported in [7] that a statistical approach such as PCA fails to encode features that are essential to the individuality of the HRTFs. Some other works such as [12, 13] proposed using fully connected auto-encoders to encode the HRTFs. A fully connected auto-encoder learns the encodings of every single HRTF in space individually, but does not consider the correlations that adjacent HRTFs share in space. Hence, as the first part of our work, we propose encoding the HRTFs by learning the correlations that HRTFs share in space using a Convolutional Neural Networks (CNN) based auto-encoder, hypothesizing that the features encoded in such a latent space would be a function of the body reflections hence making it more specific to every individual.

1.2 Problem Statement B - HRTF Interpolation

Most of the previous approaches for HRTF individualization rely on finding a mapping between the anthropometric measurements of individuals to their HRTFs. Having consumers to provide their anthropometric measurements is not necessarily a very practical solution. With the increasing availability of surround sound systems today, a rather futuristic goal would be to measure a sparse set of HRTFs with the available speakers and be able to interpolate to a complete set of HRTFs. Multiple approaches have been proposed previously to solve the problem of HRTF interpolation. Some mathematical approaches such as linear interpolation and thin plate spline interpolation have been previously employed in works such as [14–17]. Some key findings from these works suggest that thin plate spline interpolation of log magnitude HRTFs seems to achieve best interpolation results and that the number of HRTFs in space could be reduced to about 80 to allow for an accurate reconstruction. Some other works focus on studying the fitting of HRTFs onto the spatially continuous spherical harmonics (SH) domain to allow for a continuous interpolation which was first proposed by [18]. The required order of the SH increases with frequency, and the higher the order required, the more the number of HRTFs are required. It was reported in [19] that for accurate reconstruction of HRTFs upto $20kHz$, about 1600 HRTFs would be required, less than which leads to spatial aliasing which leads to high-shelf like energy increase in the SH interpolated HRTFs [20]. Some deep learning approaches such as [21, 22] have also been proposed in the past for learning HRTF interpolations, both of which deploy fully connected networks to tackle the problem. Following the success in the image world, where CNN based approaches have provided promising results for interpolation and super resolution as in [23, 24], we propose, as the second part of our work, a CNN based approach for interpolating from a few sparsely distributed HRTFs in space to to the complete HRTF set.

The remainder of the paper is structured as follows. In section 2, we briefly review the theory of HRTFs and our data preparation strategy. In section 3 we describe our proposed model architectures and the training procedures. In section 4, we describe our experiments and discuss the results for the same and finally we provide concluding statements in section 5.

2. DATA & DATA PREPARATION

2.1 Understanding HRIRs and HRTFs

For a particular point in space, the left and right HRIRs $h_l[n]$ and $h_r[n]$ are defined as the impulse responses for the acoustic system that the sound signal undergoes from its point of origination to the point it reaches the left and right ears respectively. The HRTFs $H_l(\omega)$ and $H_r(\omega)$ are the complex valued frequency domain representations for the left and right HRIRs respectively which can be represented in terms of their magnitude and phase responses as

$$H_{l/r}(\omega) = |H_{l/r}(\omega)|.\angle H_{l/r}(\omega), \tag{1}$$

where the phase response $\angle H_{l/r}(\omega)$ could be represented as,

$$\angle H_{l/r}(\omega) = e^{j\psi(\omega)} \tag{2}$$

From [25], it is possible to derive the minimum phase response $H_{min}(\omega)$ of an HRTF from its magnitude response $|H(\omega)|$. $H(\omega)$ can also be represented in terms of $H_{min}(\omega)$ as

$$H(\omega) = H_{min}(\omega).e^{j\psi_{all}(\omega)}.e^{-j\omega T}, \tag{3}$$

where $e^{\psi_{all}(\omega)}$ is an all-pass phase component and $e^{-j\omega T}$ corresponds to the propagation delay T.

Several past works [26–29] have shown that the all-pass component $e^{j\psi_{all}(\omega)}$ could be considered perceptually insignificant upto around $10kHz$. Given then that $H_{min}(\omega)$ could be obtained from $|H(\omega)|$, it is possible to estimate $H(\omega)$ from only its magnitude response and the propagation delay T. In this work, we will only focus on estimating $|H(\omega)|$. Estimation of the propagation delays is beyond the scope of this paper.

2.2 Database

We used the CIPIC database [30] for this work. The database consists of HRIRs of 45 subjects, each 200 samples long, sampled at a sampling rate of $44.1kHz$ and measured at 1250 points around the head, at a constant radius of $1m$. The positions are sampled in the form of 25 rings along the azimuth axis and at 50 points per ring along the elevation axis as shown in Fig. 1.

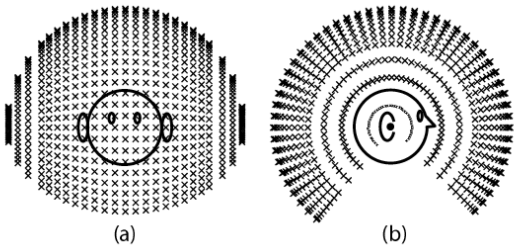


Figure 1. Sampling locations of HRTFs in the CIPIC database (a) front (b) side. The spacing between rings is 5° near the center of the head, and increases up to 15° towards the side of the head. The 50 points along every ring are equally spaced, but the spacing is larger towards the center of the head and smaller at the sides of head, to maintain a constant radius.

2.3 Data Processing and Arrangement

The HRIRs in the CIPIC database have a dynamic range of $[-2, 2]$. We first obtain the magnitude HRTFs as

$$H_{mag}(\omega) = \left| \frac{2 * FFT\left\{\frac{h[n]}{2}\right\}}{nfft} \right|, \tag{4}$$

where $nfft$, the fft length, is set to 256. $H_{mag}(\omega)$ is clipped to the minimum at 10^{-6} and we pick only the first 128 bins corresponding to positive frequencies, hence the factor of 2 to account for the energies of the negative frequencies. Equation (4) ensures that $H_{mag}(\omega)$ lies in the range of $[10^{-6}, 1]$. We then obtain the log magnitudes of the obtained magnitude HRTFs and scale them as

$$H_{log}(\omega) = \frac{log_{10}(H_{mag}(\omega))}{6} + 1, \tag{5}$$

hence ensuring that $H_{log}(\omega)$ has a dynamic range of $[0, 1]$.

The 1250 so obtained HRTF log magnitudes in the database are then arranged in a 3D tensor having a size of $[128$ x 25 x $50]$ such that the first dimension corresponds to the frequency axis and the second and the third dimensions correspond to the azimuths and the elevations respectively. In this paper to follow, we will refer to the above tensor as the HRTF-tensor.

3. PROPOSED MODEL AND TRAINING SPECIFICATIONS

3.1 The Model

3.1.1 Part A - AutoEncoder

The model architecture for our proposed auto-encoder consists of a contracting and expanding path inspired by the very successful U-NET architecture [31]. An example structure having a model depth of 3 is shown in Fig. 2. In this we first pass our input HRTF-tensor through the encoder which consists of multiple convolutional blocks starting with a convolution having a kernel size of $(3, 3)$, stride $(1, 1)$ and a padding of $(1, 1)$ ensuring that the spatial sizes are maintained (will be referred to as the size-maintaining convolutional block from her on). This is followed by several downsampling convolutions having kernel sizes of $(2, 2)$ and strides also of $(2, 2)$, with no 0 paddings. These kernels hence, both learn the features and also downsample the HRTF-tensor through multiple depth levels along the spatial dimensions. At the final depth level we apply one final size-maintaining convolution to obtain the encoded space. The achieved encoded space is then passed through the decoder which consists of a structure reversing the process carried out by the encoder. In this we start with a size-maintaining convolution followed by several transposed convolutional blocks up the depth levels, with kernel sizes, strides and paddings set appropriately, so as to make sure that the feature maps have the same spatial sizes as the corresponding feature maps of the encoder at the corresponding depth levels. Finally at depth level of 0, we have one final size-maintaining convolutional block that outputs the predicted HRTF-tensor. Just as the convs in the encoder, the transposed convs in the decoder are responsible for both learning the deep features and upsampling in the spatial dimensions. We use the exponential linear units (ELU) activations for all the convolutions and the transposed convolutions, except for the final layer in the decoder predicting the HRTFs, where we use the Sigmoid activation. We also experimented with using batch normalization, and found its effects to be negligible.

Many works have previously adopted such CNN-based auto-encoding structures for various applications, for example [32, 33]. However, our architecture differs from other similar architectures in the following ways.

First, such typical auto-encoding networks employ a number of convolutional layers at every depth level, before using pooling operations for downsampling in the encoder or after the transposed convs in the decoder. Moreover, the number of feature maps or channels double down every depth level. As previously mentioned in section 1, HRTFs come with a challenge of a small dataset, which motivates the use of as few parameters as possible to avoid overfitting. Hence, after multiple experiments, we concluded that the model works best by replacing the pooling operations by the downsampling convolutional blocks, while omitting the other convolutional layers at each depth level, hence allowing for the feature learning to be mostly carried out by the downsampling convs and the transposed convs, while also keeping the number of feature maps constant and equal to the number of frequency bins, all throughout.

Secondly, typically in 2D CNNs, each filter is expected to learn the same spatial features for all channels. We hypothesized, as also suggested in [9], that the spatial correlations of the HRTFs are expected to be different along the frequency axis. Hence we used the "groups" option in PyTorch's Conv2D function and we set it to 8. This splits the tensors along the channels axis into 8 groups before the convolution operation. Each filter now only learns the correlations of a single group, hence allowing for the learning of different features for different frequency bins. We found that this approach significantly improved the accuracy of

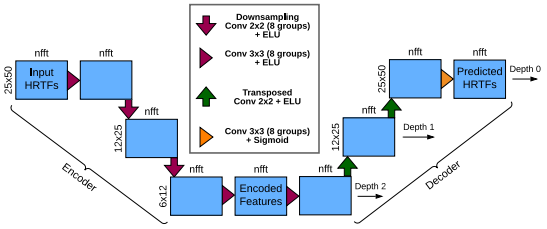


Figure 2. HRTF auto-encoder model architecture. The number above the blue blocks corresponds to the number of channels/feature maps. The number to the left of the blue blocks corresponds to the spatial size of the blocks at every depth level. The spatial sizes remain constant along every depth level. Note that nfft here corresponds to the number of positive frequency bins.

our model.

Additionally, one of the key components of a U-NET structure is the residual connections between the feature maps of the encoder to the feature maps of the corresponding depth level in the decoder. Although we concur that using residual connections does improve the accuracy of the auto-encoding structure itself, we reckon that applications of such an auto-encoding process lie in the possibility of predicting the encoded space through other means, and then using the decoder to obtain the HRTFs, an example of which we show in the work to follow. In such a case, we would not have access to the encoder during inference, and therefore we propose not using residual connections between the encoder and the decoder in this work.

3.1.2 Part B - Interpolation

For the next part of our work, we propose a transfer learning like approach for HRTF interpolation. In this, we use the learned decoder from the auto-encoder network and train a new encoder (will be referred to as the sparse-encoder from here on) to map the sparse HRTF-tensor onto the encoded space obtained using the auto-encoder. The flow diagram for our model is shown in Fig. 3. The encoding network for the sparse-encoder comprises of a single convolutional layer, parameters of which are decided by the spatial size of the input sparse HRTF-tensor that we wish to interpolate. In this work, we demonstrate the interpolation for two cases – 1. that of a sparse HRTF-tensor having a spatial size of 6 x 12, and 2. that of a sparse HRTF-tensor having size 3 x 6.

For case 1 we use for the encoder, a single convolutional layer with kernel size $(3, 3)$, stride $(1, 1)$ and padding of $(1, 1)$. For case 2 we use a single transposed convolution layer with kernel size $(2, 2)$ and stride of $(2, 2)$ with no 0 padding. We use the ELU activation for both cases since the decoder was trained to decode from an encoded space which had gone through an ELU activation itself.

The decoder structure for both cases is the same as that of the auto-encoder with a model depth of 3.

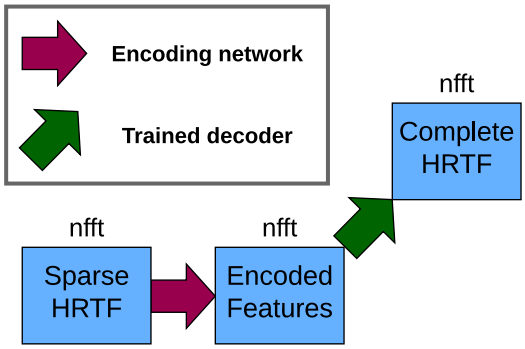


Figure 3. Flow diagram for interpolation of sparse HRTF

3.2 Training Specifications

Of the 45 subjects available in the CIPIC database, we randomly set 3 subjects aside as test subjects and trained our models on the remaining 42 subjects. For both the auto-encoding and the interpolation networks, we use full-batch training using the ADAM optimizer with an initial learning rate of 0.0001. We use the early stopping strategy where we let our training process run for a maximum of 400000 epochs and stop the training when the validation error starts to go up while picking the model with the lowest validation loss. More sophisticated validation strategies such as leave-one-out cross validation, which is often suited for small datasets, but which requires significant resources in terms of the total training time, remain to be a future work at the moment.

For the interpolation problem, we first trained the sparse-encoder while freezing the weights of the pre-trained decoder from the autoencoder model, until the validation error reached a plateau. We then unfroze the weights of the decoder while setting the learning rate of the decoder to $\frac{1}{100}^{th}$ that of the sparse-encoder and continued the training process following the early stopping procedure. We found that this scheme of fine-tuning the weights of the pre-trained decoder to provide better results as opposed to training the decoder and the sparse encoder end-to-end, or as opposed to not fine-tuning the pre-trained decoder at all.

After experimentation, we found that use of dropouts and regularization schemes to not be very effective for these experiments and they only prolonged the training process, without actually resulting in better reconstructions.

3.3 Loss Function

We used a mean Log Spectral Distortion (LSD) function as both the loss function to train our models and the evaluation criteria. For a pair of ground truth and predicted log magnitude HRTFs H and $\hat{H}$ respectively, the LSD can be defined as

$$LSD(H, \hat{H}) = \sqrt{\frac{1}{k_2 - k_1 + 1} \sum_{k=k_1}^{k_2} \left(\hat{H}(k) - H(k)\right)^2}, \quad (6)$$

where the frequency bin numbers k_1 and k_2 correspond to the starting and ending frequency bins respectively along which to compute the LSD. We used full band LSD to evaluate our models, in other words setting k_1 to 0 and k_2 to 127.

4. EXPERIMENTS, RESULTS & DISCUSSION

Before moving on to the experiments, we would like to give a few details about the LSD plots that we use to report our results.

4.1 Understanding the LSD Plots

The LSD plot demonstrates the LSD values of the reconstructed HRTFs for a particular subject at all points in space available in the database, in our case 1250 points. We use the inter-aural polar co-ordinate system to label the azimuth θ and the elevation ϕ angles. In this, the points to the left of the head are found at azimuths $80° > \theta > 0°$ whereas points to the right of the head are found at $0° > \theta > -80°$ with $\theta = 0°$ being exactly in front or back of the head. Points in front of the head are found at elevations $-45° < \phi < 90°$ while points at the back of the head are found at $90° < \phi < 230°$ with $\phi = 0°$ and $\phi = 180°$ corresponding to the lateral plane at ear level and $\phi = 90°$ being the point exactly above the head. Thereby, $-45° < \phi < 0°$ and $230° > \phi > 180°$ correspond to points below the head and $0° < \phi < 180°$ correspond to points above the head.

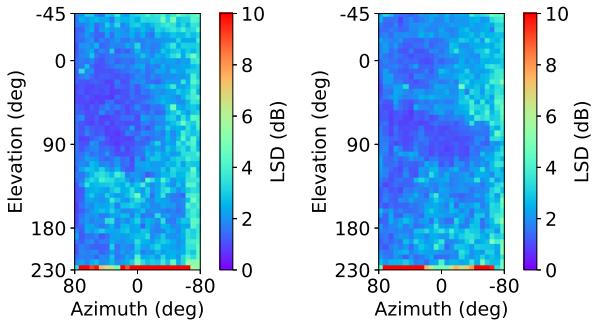
In the remainder of this section, we report the results of our auto-encoder and interpolation models. For brevity, we will only be reporting results for the predictions of the log magnitude HRTFs of the left ear.

4.2 Experiment 1 - AutoEncoding

The aim of this experiment was to study the reconstruction capability of our auto-encoder model described in section 3.1.1 for model depths of 3 and 4. A model depth of 3 encodes the HRTF-tensor, which consists of 25x50 HRTFs, to an encoded space having a spatial size of 6x12 and as many channels as the number of frequency bins, in our case 128. A model depth of 4 encodes the HRTF-tensor one step further to an encoded space of size 128x3x6.

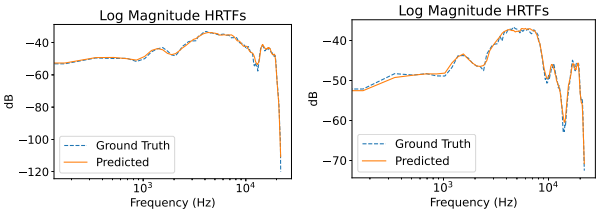
Fig. 4 shows the LSD plot for 2 test subjects for a model depth of 3. Fig. 5 compares the reconstructed HRTFs with the ground truth HRTFs for the point located at [$\theta = 45°$, $\phi = 45°$], or in other words the point at $45°$ to the left and above in front of the head.

The LSD plots suggest that the reconstruction is better in the area above the head (the upper half of the lateral plane) and on the ipsilateral side of the HRTFs (in this case the left side for left HRTFs). There is a significant drop in the performance for points on the extreme bottom behind the head and also at some points on the extreme right side. We suspect this error to be related to either the 0 padding schemes in our convolutional layers, or bad points in the dataset itself, and further investigation on this anomaly is left for future work.



(a) Subject 0 (b) Subject 1

Figure 4. LSD plots for the AutoEncoder predictions for a model depth of 3 for 2 test subjects (left ear)

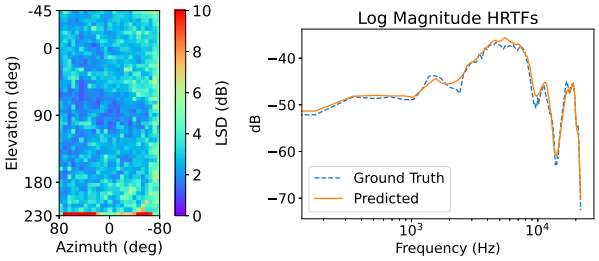


(a) Subject 0 (b) Subject 1

Figure 5. HRTF plots for the AutoEncoder predictions for a model depth of 3 for 2 test subjects at an azimuth and elevation of $45°$ on the ipsilateral side of the left ear

The HRTF plots reveal that although the HRTFs for the two subjects at the same location differ significantly from one another, the model was able to reconstruct these to a great degree, hence indicating that it successfully encodes individuality in the HRTFs.

Fig. 6 shows the LSD plot and the HRTF plot for 1 subject for a model depth of 4. As expected, the reconstruction errors are higher in this case as a result of encoding the HRTF-tensor to an even smaller space. The HRTF plot suggests that this results in a rather more smoothed HRTF reconstruction. Although, whether this difference in the reconstruction is audible could only be answered through perceptual listening tests, which is left for future work. Table 1 compares the mean LSD values and their standard deviations across all 3 test subjects and for all points in space for the two cases. Interestingly, the standard deviations for both models are exactly the same. This suggests that the correlations between certain points in space are being better learned than at other points, irrespective of the model depth. Given that CNNs assume an equivalent correlation across all points in space, splitting the HRTF-tensors into multiple regions spatially and applying a different network to each region might provide better results and is left for future work. Furthermore, it might also be worthwhile to obtain results on some other HRTF databases where the points are sampled at spherically equidistant locations such as the ARI database [34].



(a) LSD plot (b) HRTF plot

Figure 6. LSD and HRTF plots for AutoEncoder with a model depth of 4 (left ear)

Depth Level	Mean ± Std LSD (dB)
3	2.38 ± 1.52
4	3.24 ± 1.52

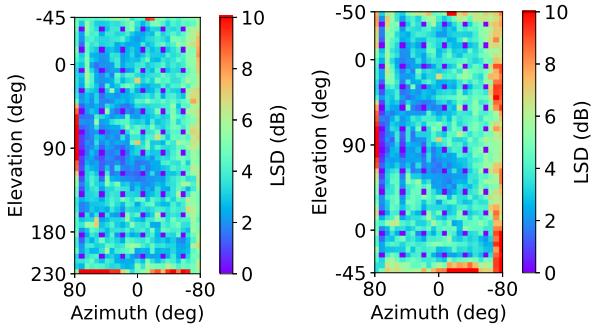
Table 1. Comparing mean and std LSD values for AutoEncoder with model depths 3 v/s 4. Note that the standard deviations are calculated across HRTFs at all positions in space and for all the test subjects

4.3 Experiment 2 - Interpolation

To obtain the sparse HRTF-tensor, we sampled the complete HRTF-tensor uniformly. We obtained 2 such sparse HRTF-tensors. For the first one, we sampled at every 4 points in both the azimuth and elevation dimensions, starting from index 1 for azimuth ($[1, 5, 9...]$) and starting from index 2 for the elevations ($[2, 6, 10, ...]$), thereby obtaining a sparse HRTF-tensor having a size of 6x12. For the second one, we sampled at every 8^{th} point starting at index 4 in both dimensions ($[4, 12, ...]$), thereby obtaining a sparse HRTF-tensor with size 3x6. We will refer to these sparse HRTF-tensors as sparse-tensor-1 and sparse-tensor-2 respectively for the discussion to follow. After obtaining our predicted complete HRTF-tensors, we re-inserted the corresponding sparse HRTF-tensors in the predictions at the appropriate positions.

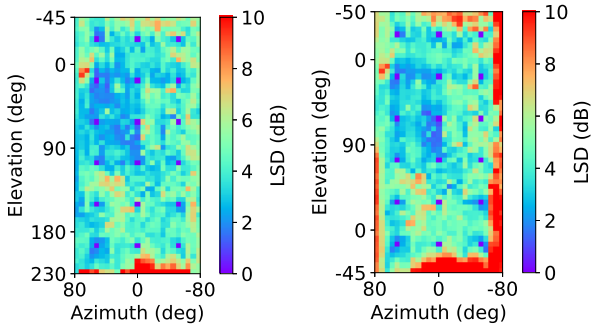
To report our results, we compare our proposed interpolation strategy with that obtained using bilinear interpolation. Bilinear interpolation only allows for interpolation between points inside the boundaries of the sparse-tensor and is incapable of extrapolating points outside the boundaries. However, the CNN model provides both interpolation as well as extrapolation. For visualization purposes, in the bilinear interpolation scheme, we copied the HRTFs at the boundary to substitute for the unavailable points outside the boundaries.

Figs. 7 and 8 show the LSD plots for the interpolations of sparse-tensor-1 and sparse-tensor-2 respectively. We can see some improvement in our proposed model with respect to the bilinear interpolation scheme, especially in the region above the head and on the ipsilateral side. The improvement is more apparent for sparse-tensor-2. We can also see that the method does a fair job in extrapolating points outside the selected region in the front of the head and on the ipsilateral side. We achieved a mean LSD



(a) Proposed Model (b) Bilinear interpolation

Figure 7. LSD plots for the interpolation of sparse-tensor-1 (left ear)



(a) Proposed Model (b) Bilinear interpolation

Figure 8. LSD plots for the interpolation of sparse-tensor-2 (left ear)

of $3.34dB$ and $4.12dB$ for the interpolation and extrapolation of sparse-tensor-1 and sparse-tensor-2 respectively. Table 2 provides the LSD comparisons for all test subjects over all the locations between the proposed model and the bilinear interpolation scheme. To allow for a fair comparison, we only report the LSDs for the interpolation of points inside the boundary in this table. The LSD results show that the proposed model provides about 10% improvement over the bilinear interpolation scheme.

Model	Sparse-tensor-1 Mean ± Std LSD	Sparse-tensor-2 Mean ± Std LSD
CNN	2.82 ± 1.29	3.61 ± 1.32
Bilinear	2.99 ± 1.37	3.95 ± 1.39

Table 2. Comparing the mean and std LSD values for interpolation by the proposed CNN model v/s bilinear interpolation

5. CONCLUSION

In this work, we presented an alternate strategy for the encoding of HRTFs using a CNN-based auto-encoder. We showed that an HRTF-set consisting of 1250 locations could be encoded to a latent space having a spatial size

of 6x12 with a mean LSD of $2.38dB$ and to a space having a size of 3x6 with a mean LSD of $3.24dB$. We further showed that such an encoded space could be used towards learning interpolations from an HRTF-set sparsely sampled in space. In doing so, we showed that a sparse HRTF-set sampled at 72 locations results in a mean reconstruction error of $3.34dB$ LSD and that sampled at 18 locations results in a mean reconstruction error of $4.12dB$ LSD, both cases also supported by the LSD plots providing more insight into the LSD errors per location. The results also suggested that such an interpolation scheme allows for not only interpolation, but also extrapolation to some extent.

Some immediate next steps in this work remain to be trying this algorithm on other available datasets to obtain a more generalized result, while also using more sophisticated cross-validation strategies such as the leave-one-out strategy. Finally, it would be crucial to conduct perceptual listening tests to understand the perceptual effects of the proposed approach.

Having said that, the above presented results provide plenty insights into suggesting that it is indeed possible to encode HRTFs using convolutional neural networks using only 42 datapoints while allowing the encoding of features essential for the individuality of the HRTFs. This leads way to finding mappings between other features that define the HRTFs and the learned encoded space, for example, using using sparsely sampled HRTFs in space as shown in this work. We showed that the encoded space could be leveraged to interpolate HRTFs to an acceptable degree with only 18 locations. With the extent of surround sound systems that are available today such as the 11.1.8 system which consists of 19 speakers, we believe as a rather futuristic goal, that this work might be the first step into finding mappings between the latent representations of the HRTFs measured using these speakers (although corrupted with room reflections) to the latent space of the user's clean and complete HRTFs, allowing for the prediction of the complete clean HRTF set using the corrupted sparse HRTF measurements obtained using the surround sound speakers for example. We hope that such mappings between latent features of similar acoustic systems might open up interesting avenues.

6. REFERENCES

[1] E. M. Wenzel, M. Arruda, D. J. Kistler, and F. L. Wightman, "Localization using nonindividualized head-related transfer functions," *The Journal of the Acoustical Society of America*, vol. 94, no. 1, pp. 111–123, 1993.

[2] C. Guezenoc and R. Seguier, "Hrtf individualization: A survey," *arXiv preprint arXiv:2003.06183*, 2020.

[3] D. Zotkin, J. Hwang, R. Duraiswaini, and L. S. Davis, "Hrtf personalization using anthropometric measurements," in *2003 IEEE workshop on applications of signal processing to audio and acoustics (IEEE Cat. No. 03TH8684)*. Ieee, 2003, pp. 157–160.

[4] P. Bilinski, J. Ahrens, M. R. Thomas, I. J. Tashev, and J. C. Platt, "Hrtf magnitude synthesis via sparse representation of anthropometric features," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4468–4472.

[5] R. Miccini and S. Spagnol, "Hrtf individualization using deep learning," in *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, 2020, pp. 390–395.

[6] C. J. Chun, J. M. Moon, G. W. Lee, N. K. Kim, and H. K. Kim, "Deep neural network based hrtf personalization using anthropometric measurements," in *Audio Engineering Society Convention 143*. Audio Engineering Society, 2017.

[7] H. Fayek, L. van der Maaten, G. Romigh, and R. Mehra, "On data-driven approaches to head-related-transfer function personalization," in *Audio Engineering Society Convention 143*. Audio Engineering Society, 2017.

[8] G. W. Lee and H. K. Kim, "Personalized hrtf modeling based on deep neural network using anthropometric measurements and images of the ear," *Applied Sciences*, vol. 8, no. 11, p. 2180, 2018.

[9] K. Yamamoto and T. Igarashi, "Fully perceptual-based 3d spatial sound individualization with an adaptive variational autoencoder," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–13, 2017.

[10] Y. Shu-Nung, T. Collins, and C. Liang, "Head-related transfer function selection using neural networks," *Archives of Acoustics*, vol. 42, no. 3, pp. 365–373, 2017.

[11] H. Hu, L. Zhou, H. Ma, and Z. Wu, "Hrtf personalization based on artificial neural network in individual virtual auditory space," *Applied Acoustics*, vol. 69, no. 2, pp. 163–172, 2008.

[12] T.-Y. Chen, T.-H. Kuo, and T.-S. Chi, "Autoencoding hrtfs for dnn based hrtf personalization using anthropometric features," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 271–275.

[13] Y. Luo, D. N. Zotkin, and R. Duraiswami, "Virtual autoencoder based recommendation system for individualizing head-related transfer functions," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013, pp. 1–4.

[14] K. Hartung, J. Braasch, and S. J. Sterbing, "Comparison of different methods for the interpolation of head-related transfer functions," in *Audio Engineering Society Conference: 16th International Conference: Spatial Sound Reproduction*. Audio Engineering Society, 1999.

[15] S. Carlile, C. Jin, and V. Van Raad, "Continuous virtual auditory space using hrtf interpolation: Acoustic and psychophysical errors," in *Proceedings of the First IEEE Pacific-Rim Conference on Multimedia*, 2000, pp. 220–223.

[16] P. Minnaar, J. Plogsties, and F. Christensen, "Directional resolution of head-related transfer functions required in binaural synthesis," *Journal of the Audio Engineering Society*, vol. 53, no. 10, pp. 919–929, 2005.

[17] R. Martin and K. McAnally, "Interpolation of head-related transfer functions," DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION EDINBURGH (AUSTRALIA) AIR . . . , Tech. Rep., 2007.

[18] M. J. Evans, J. A. Angus, and A. I. Tew, "Analyzing head-related transfer function measurements using surface spherical harmonics," *The Journal of the Acoustical Society of America*, vol. 104, no. 4, pp. 2400–2411, 1998.

[19] W. Zhang, T. D. Abhayapala, R. A. Kennedy, and R. Duraiswami, "Insights into head-related transfer function: Spatial dimensionality and continuous representation," *The Journal of the Acoustical Society of America*, vol. 127, no. 4, pp. 2347–2357, 2010.

[20] J. M. Arend, F. Brinkmann, and C. Pörschmann, "Assessing spherical harmonics interpolation of time-aligned head-related transfer functions," *Journal of the Audio Engineering Society*, vol. 69, no. 1/2, pp. 104–117, 2021.

[21] V. Lemaire, F. Clerot, S. Busson, R. Nicol, and V. Choqueuse, "Individualized hrtfs from few measurements: a statistical learning approach," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4. IEEE, 2005, pp. 2041–2046.

[22] G. Kestler, S. Yadegari, and D. Nahamoo, "Head related impulse response interpolation and extrapolation using deep belief networks," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 266–270.

[23] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[24] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[25] A. Oppenheim, R. Schafer, and J. Buck, "Discrete-time signal processing: Prentice-hall englewood cliffs," 1989.

[26] S. Mehrgardt and V. Mellert, "Transformation characteristics of the external human ear," *The Journal of the Acoustical Society of America*, vol. 61, no. 6, pp. 1567–1576, 1977.

[27] A. Kulkarni, S. Isabelle, and H. Colburn, "On the minimum-phase approximation of head-related transfer functions," in *Proceedings of 1995 Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 1995, pp. 84–87.

[28] D. J. Kistler and F. L. Wightman, "A model of head-related transfer functions based on principal components analysis and minimum-phase reconstruction," *The Journal of the Acoustical Society of America*, vol. 91, no. 3, pp. 1637–1647, 1992.

[29] A. Kulkarni, S. Isabelle, and H. Colburn, "Sensitivity of human subjects to head-related transfer-function phase spectra," *The Journal of the Acoustical Society of America*, vol. 105, no. 5, pp. 2821–2840, 1999.

[30] V. R. Algazi, R. O. Duda, D. M. Thompson, and C. Avendano, "The cipic hrtf database," in *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No. 01TH8575)*. IEEE, 2001, pp. 99–102.

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[32] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2016, pp. 424–432.

[33] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "Wgansing: A multi-voice singing voice synthesizer based on the wasserstein-gan," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[34] Ari database webpage. [Online]. Available: https://www.oeaw.ac.at/en/isf/das-institut/software/hrtf-database

A Differentiable Neural Network Approach To Parameter Estimation Of Reverberation

Søren Vøgg Lyster
Aalborg University Copenhagen
slyste20@student.aau.dk

Cumhur Erkut
Aalborg University Copenhagen
cer@create.aau.dk

ABSTRACT

Differentiable Digital Signal Processing is a library and set of machine learning tools that disentangle the loudness and pitch of an audio signal for timbre transfer or for applying digital audio effects. This paper presents a DDSP-based neural network that incorporates a feedback delay network plugin written in JUCE in an audio processing layer, with the purpose of tuning a large set of reverberator parameters to emulate the reverb of a target audio signal. We first describe the implementation of the proposed network, together with its multiscale loss. We then report two experiments that try to tune the reverberator plugin: a "dark" reverb where the filters are set to cut frequencies in the middle and high range, and a "brighter", more metallic sounding reverb with less damping. We conclude with the observations about advantages and shortcomings of the neural network.

1. INTRODUCTION

Designing a good sounding artificial reverberator is a hard task [1]. An algorithm with multiple filters and delay lines can consist of a high number of adjustable parameters, and the task of tuning these parameters by hand and ears to achieve the desired reverberation can take hours or days, even for a skilled audio engineer. Estimating a large number of parameters to reach a desired target is a use case that fits well into the subject of machine learning and neural networks. We present an adaptation of a neural network model to estimate a large set of parameters of a reverberator with the purpose of tuning that reverberator to emulate a target reverberated audio signal.

Google's Magenta Team recently released the Differentiable Digital Signal Processing (DDSP) [2] and has presented an approach to neural networking that incorporates audio processing in a neural network model and a multiscale spectral loss for audio differentiation. The loss can then be propagated back through the model to update the weights and change the audio generative parameters until the output signal resembles the target signal. The most prominent use of DDSP is to train a recurrent neural network on audio of an instrument for it to be able to infer the timbre information of the given instrument and transfer it to another audio input [1].

With the introduction of DDSP more research has gone into incorporating different digital signal processing elements in a differentiable neural network. Kuznetsov and his colleagues presented multiple approaches to tuning filters using a differential neural network [3], and they have encountered the issue of training neural networks when a DSP element has a high probability of being unstable, e.g., when selecting coefficients for a biquad filter.

Extending the work of DDSP Ramírez et al. [4] propose using the differential signal processing to estimate parameters of third-party black box audio effect plugins by incorporating them in the neural network. Their work has been done on the LV2 audio plugin framework, and cases was shown for tube amplifier emulation, artifact removal from voiced signals, and music mastering.

Spotify's artificial intelligence research and development department Audio Intelligence [2] has released a utility framework for hosting Virtual Studio Technology audio plugins (VST) in Python called Pedalboard [5]. This utility eases the use of integrating black box audio plugins in a machine learning and neural network environment such as Tensorflow [6].

Our contribution to this body of work are as follows:

- Replacing the biquads of [3] with the state-variable filters (SVFs) for stability,
- Applying the two gradient estimation methods of [4], namely the finite differences and simultaneous perturbation stochastic approximation, to the SVFs and highly recurrent structures of Feedback Delay Networks (FDN) [7], and
- Establishing a building block in differentiable artificial reverberation [8] between black-box and white-box approaches.

2. DESIGN

To create a neural network that can estimate reverberator parameters by audio differentiation we need a reverberator plugin that can be incorporated in the network structure. Instead of finding an open-source reverberator we opted to create a custom audio plugin since it would present the easiest way to define exactly what parameters to expose to the neural network.

Copyright: © 2022 Søren Vøgg Lyster et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[1] https://sites.research.google/tonetransfer
[2] https://research.atspotify.com/audio-intelligence/

2.1 Feedback Delay Network

For the reverberator a feedback delay network (FDN) was selected since it generally should be able to emulate a broad selection of different reverberator algorithms [9] [1]. The FDN is an algorithm emulating reverb by having multiple delay lines with feedback through a scattering matrix [1]. A correct selection of delay lengths and scattering matrix should be able to result in a lossless algorithm that will produce an approximation to white noise when applied with an impulse [10].

The equations for the general FDN are shown in the following two equations, where x is the input, y is the output, and s_i is the delayed signal state at i, with i being the index in the number of delay lines N. A is the $N \times N$ scattering feedback matrix with indexes i and j. The variables c and b are gain factors in the system. The delay in samples n for each delay line is defined as M.

$$y(n) = \sum_{i=1}^{N} c_i \, s_i(n) \tag{1}$$

$$s_i(n + M_i) = \sum_{j=1}^{N} A_{ij} s_j(n) + b_i x(n) \tag{2}$$

Many different solutions for implementing a feedback matrix exists [10], but for now the matrix A is selected to be a lossless Householder matrix. The matrix for a 4-channel FDN implementation A_4 can be seen in equation 3, while an extension into a 16 channel implementation A_{16} can be seen in equation 4.

$$A_4 = \frac{1}{2} \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix} \tag{3}$$

$$A_{16} = \frac{1}{2} \begin{bmatrix} A_4 & -A_4 & -A_4 & -A_4 \\ -A_4 & A_4 & -A_4 & -A_4 \\ -A_4 & -A_4 & A_4 & -A_4 \\ -A_4 & -A_4 & -A_4 & A_4 \end{bmatrix} \tag{4}$$

After achieving a lossless reverb, the FDN algorithm can be extended to include frequency dependent dampening with the introduction of filters in the delay lines. Initially these filters were biquads, but they were later changed for state-variable filters (SVF) [11] due to their performance when calculating gradients. State variable filters can be described as a series of bi-quad filters consisting of a low-pass, a band-pass, and a high-pass filter. The benefits of SVF over normal biquads are the easy stability conditions of the contained filters, since they do not directly utilize poles and zeros, but instead have a dampening coefficient R and frequency coefficient g [3]. The stability conditions for the SVF are $R > 0$ and $g > 0$. The equations describing the input x and output y relationship and the internal states h_1 and h_2 (initialized at 0) of the SVF can be seen in equation 5 through 10, where y_{LP}, y_{BP}, and y_{HP} are the output of each filter with c_{LP}, c_{BP}, and c_{HP} being their respective gains.

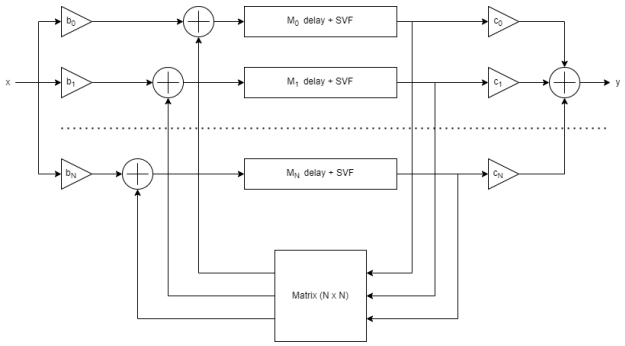


Figure 1: A diagram of the FDN algorithm implemented, showing delay lines $1, 2, \ldots, N$ with M_N samples delay and SVF filters, gain factors b_N and c_N, and the $N \times N$ scattering feedback matrix.

$$y_{BP}[n] = \frac{g([n] - h_2[n-1]) + h_1[n-1]}{1 + g(g + 2R)} \tag{5}$$

$$y_{LP}[n] = g \, y_{BP}[n] + h_2[n-1] \tag{6}$$

$$y_{HP}[n] = x[n] - y_{LP}[n] - 2R y_{BP}[n] \tag{7}$$

$$h_1[n] = 2y_{BP} - h_1[n-1] \tag{8}$$

$$h_2[n] = 2y_{LP} - h_2[n-1] \tag{9}$$

$$y[n] = c_{HP} \, y_{HP} + c_{BP} \, y_{BP} + c_{LP} \, y_{LP} \tag{10}$$

The value of the delay lengths M should be mutually prime to encourage denser echoes and avoid artifacts [10]. The delay lengths should be parameterized and exposed to the neural network but that would require that they are handled by some higher-level parameter since the network is unlikely to estimate the values to be mutually prime, as shown in earlier experiments. For this project the delay lengths have been fixed to a set of values given in the implementation of Prawda et al.'s highly flexible feedback delay network [9]. A diagram of the FDN reverberator can be seen in figure 1.

This FDN algorithm can be expressed as having 7 parameters for each delay line i: b_i, c_i, C_{HPi}, C_{BPi}, C_{LPi}, g_i, and R_i. Running a configuration with 16 delay lines results in 112 exposed parameters.

2.2 Neural Network

To achieve the set-out goals we need a neural network that can take a given input tensor containing audio and then output a tensor to compare with the target audio. The network should be able to update its weights by backwards propagation of gradients and loss. To incorporate the instances of the custom reverberator VST in the neural network model a custom audio processing layer needs to be implemented, and to propagate gradients back through the network the custom layer needs to implement a custom gradient function. This section will describe the different elements of the neural network model. A diagram of the neural network model can be seen in figure 2.

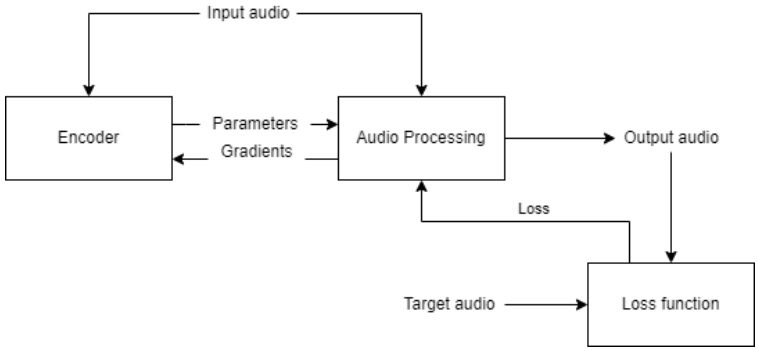


Figure 2: The implemented neural network model consisting of the encoder estimating parameters for the audio processing layer, the audio processing layer hosting the reverberator plugin, and the loss function returning a loss value that is propagated back through the model with the gradients that are calculated for the parameters in the audio processing layer.

2.2.1 Gradient method

To be able to do the full back propagation through the model the audio processing layer needs to estimate a gradient vector. Ramírez et al. [4] proposes two different ways of doing this: finite differences and simultaneous perturbation stochastic approximation. Both gradient estimation methods are implemented so they can be evaluated for this project.

The stochastic gradient estimation with finite differences (FD) [12] creates the gradient vector by iterating through all the parameters $\hat{\theta}$ and estimating the signal difference $\widetilde{\nabla}^{FD} f(\hat{\theta})$, where $f(.)$ is the neural network output driven by the dry audio signal. The equation for estimating a single parameter $\hat{\theta}_i$ with a fixed value ϵ and a standard basis of size equal to the number of parameters d^P indexed at the parameter index i is shown in equation 11

$$\widetilde{\nabla}^{FD} f(\hat{\theta})_i = \frac{f(\hat{\theta} + \epsilon d_i^P) - f(\hat{\theta} - \epsilon d_i^P)}{2\epsilon}. \tag{11}$$

This finite difference requires the plugin to generate audio twice for each parameter, with the parameter $\theta_i + \epsilon$ and $\theta_i - \epsilon$, and an additional time to create an output signal. This results in $2 \times P + 1$ computations for each batch item per epoch (with P being the number of parameters). For a reverberator with a long impulse response tail, this approach may be not feasible; it is used here however as a benchmark. Still, it is only used in the training phase and does not have any computational demands on the reverbarator in run-time.

Another approach to gradient estimation is the simultaneous perturbation stochastic approximation (SPSA). SPSA is encouraged when dealing with a high-dimensional problem [13] [12] and is done by creating a P-sized vector of perturbations Δ^P that contains differences for each parameter. In this case the vector contains -1 and 1 from a binomial distribution with p-value 0.5. The SPSA approach only requires $2 + 1$ computations compared to the $2P + 1$ computations of the FD method. The gradient calculation for a parameter with index i can be seen in equation 12.

$$\widetilde{\nabla}^{SPSA} f(\hat{\theta})_i = \frac{f(\hat{\theta} + \epsilon \Delta^P) - f(\hat{\theta} - \epsilon \Delta^P)}{2\epsilon \Delta_i^P} \tag{12}$$

2.2.2 Encoder

The task of the encoder is to encode an audio input into a set of values equal the number of parameters exposed by the reverberator plugin. Multiple different encoder models could be utilized for this task, but for simplicity the MobileNetV2 is selected as in the DeepAFx implementation [14] [4]. The MobileNetV2 is a two-dimensional convolutional network, so we use a logarithmic Mel Spectrogram layer to convert the input audio before the MobileNetV2.

2.2.3 Custom audio processing layer

The custom audio processing layer should receive an input the size of the number of parameters. During training these parameters will be used to generate gradients using the FD or SPSA method. When building the layer an instance of the VST plugin is created for each item in the given batch. This allows us to utilize the parallelism of Tensorflow and speed up the training. The VST plugin generates the entire audio signal in a single run, and its internal state is reset between each forward run. When training and calculating the vector containing the P number of parameter gradients the plugin is run either $P \times 2 + 1$ times (FD) or $P + 1$ times (SPSA), with the internal state being reset between each run.

2.2.4 Loss function

The loss function is an adaptation of the Multi-Scale Spectral Loss implementation from DeepAFx [4]. The short time Fourier transformation S is calculated across the audio signal for each window size i in one of $[1024, 512, 256]$, where y is the target audio and $\hat{y}$ is the output audio. The loss sum for each window size L_i is comprised of a L2 magnitude loss, the logarithmic of the L2 magnitude loss, and a L1 magnitude loss and can be seen in 13. The current construction of the loss is selected by iteratively testing the model training performance.

$$\begin{aligned} L_i = \|S_i(\hat{y}) - S_i(y)\|_2 + \log(\|S_i(\hat{y}) - S_i(y)\|_2) \\ + \|S_i(\hat{y}) - S_i(y)\|_1 \end{aligned} \tag{13}$$

The final loss value L_{tot} is a summation of different losses across the window size range N (equation 14).

$$L_{tot} = \sum_{i=0}^{N} L_i \tag{14}$$

2.2.5 Optimizer

As optimizer the Adam algorithm is chosen. The Adam algorithm is a modification of the stochastic gradient descent method that is easy to use and common in many newer neural networks such as DeepAFx [4] and DDSP [2].

2.2.6 Hyper-parameters

Selecting appropriate hyper-parameters for the gradient estimation and optimizer has been done by multiple iterations and evaluations of the model. Selecting a correct learning rate has been crucial to achieve a converging model that does not get stuck on local minimums. An initial high learning rate of 1×10^{-3} has been set with a callback for iterative reduction of the learning rate to 1.6×10^{-6} when the loss plateaus over a set number of epochs. For the gradient estimation the value ϵ has been set to 0.05.

3. IMPLEMENTATION

3.1 Reverberator

The FDN reverberator is implemented using JUCE [3] . JUCE is a c++ programming framework that is highly suitable for developing audio processing plugins. The FDN reverberator is compiled as a VST3 plugin using the VST3 SDK from Steinberg [4] . Using GitHub's continuous integration and development tool GitHub Actions an automated workflow has been utilized to compile the plugin for x86_64 Ubuntu 18.04 and publish to the neural network environment. The code is available on GitHub [5] .

The gain variables c and b, and the SVF filter variables C_{LP}, C_{BP}, C_{HP}, g, and R for each delay line are exposed as parameters using the JUCE AudioProcessorValueTreeState class. This allows the neural network to tweak those parameters. The variables M for delay length, and A for the scattering feedback matrix are not exposed. Since the delay lengths benefit greatly from being mutual primes and possibly span a large range of integers, the task of estimating them directly was deemed inefficient for the neural network. The scattering feedback matrix was chosen to be unchangeable since it easily could cause instability in the system. An important step in the implementation has also been to assure that the VST3 plugin could run headless, since we do not want to instantiate a graphical user interface for each forward processing run.

3.2 Neural Network

A repository containing the implemented neural network can also be found at Github [6] .

3.2.1 Model architecture

Two models have been implemented in Tensorflow with Keras, where the first model is the decoder model estimating parameter values from an audio input seen in Table 1, and the second model is the full differential model consisting of the decoder model and the custom VST3 layer seen in Table 2.

The log mel spectrogram layer implementation is taken from Keunwoo Choi [7] . The multi-scale spectral loss function is adapted from the DeepAFx Github repository [8] .

3.2.2 Gradient estimation

When running the model all 112 initial parameters are set to a value of 0.5. Unfortunately, the SPSA method does not seem to allow the model to converge in the same manner as FD. One reason that FD performs better than SPSA for the FDN plugin might be the result of the SVF parameters being highly dependent on each other, and that the effect of changes in the 16 parallel filters are hard to distinct when calculating gradients using this approach. This effect becomes more apparent when comparing the implementation to a simplified toy model with only one filter, where SPSA performs well.

3.3 Python Integration Using Pedalboard

Spotify's Audio Intelligence Lab has created a Python package called Pedalboard [5] that supports hosting of VST3 and Audio Unit plugins in python. This is done by implementing Python Bindings that allows for C and C++ integration in Python. The package contains functionality to process audio, change parameters, and reset the internal state of plugins (given that the plugin supports these functions). The package also claims to be thread-safe, supporting multiple CPU cores, and compatible with tensor inputs. Utilizing this package has made the implementation of the custom reverberator plugin in a neural network possible.

3.4 Toy Model

A simplified FDN plugin with only one delay line has been implemented to test the neural network model. This simplified version consisted of only 7 parameters, greatly reducing the complexity of the estimation problem. The toy model implementation showed that a higher ϵ value used in the gradient approximation would result in a faster convergence, but with the drawback of possible tuning the SVF parameters quickly towards instability and resulting in exploding gradients. Two short experiments were done to verify the ability of the neural network using two target audio sources generated with the simplified FDN implementation, a signal with high-frequency attenuation ("dark") and a signal with low frequency attenuation ("bright"). The result is reported on in table 3.

[3] https://juce.com/
[4] https://www.steinberg.net/developers/
[5] https://github.com/VoggLyster/Reverberator/tree/SMC
[6] https://github.com/VoggLyster/ReverberatorEstimator/tree/SMC
[7] https://gist.github.com/keunwoochoi/f4854acb68acf791a49a051893bcd23b
[8] https://github.com/adobe-research/DeepAFx/blob/main/deepafx/losses.py

Layer (type)	**Output Shape**	**Param #**
audio_input (InputLayer)	[(None, 96000)]	0
log_melgram_layer (LogMelgramLayer)	(None, 372, 128, 1)	0
input_norm (BatchNormalizationLayer)	(None, 372, 128, 1)	4
mobilenetv2_1.00_372 (Funtional)	(None, 112)	2400880

Total params: 2,400,884
Trainable params: 2,366,770
Non-trainable params: 34,114

Table 1: Parameter model summary from Tensorflow.

Layer (type)	**Output Shape**	**Param #**	**Connected**
audio_input (InputLayer)	[(None, 96000)]	0	
parameter_model (Functional)	(None, 112)	2400884	audio_input[0][0]
vst_processor (VSTProcessor)	(None)	0	audio_input[0][0] parameter_model[0][0]

Total params: 2,400,884
Trainable params: 2,366,770
Non-trainable params: 34,114

Table 2: Full model summary from Tensorflow.

	loss	mae
1. "dark"	0.11432	1.9732e-4
2. "bright"	0.0122	7.834e-5

Table 3: Error values for two toy model experiments

	loss	mae	mse
1. "dark"	3.12275	0.0111	0.0007
2. "bright"	3.92548	0.0115	0.0010

Table 4: Error values for the two experiments

4. EXPERIMENTS

To test the model two final experiments were done. Both experiments consisted of a two second audio input and target pair at 48kHz. These pairs where created using the implemented FDN reverberator plugin to ensure that a target solution exists where the algorithms used are able to emulate each-other. The first target is a "dark" reverb where the filters are set to cut frequencies in the middle and high range. The other target is a "bright" and more metallic sounding reverb with less damping. For generating the sound an impulse-like finger snap sample has been used.

To utilize the multiple GPU's available for training a batch size of 8 has been chosen. The batch consists of 8 input audio tensors with the same generated input audio sample, and 8 target audio tensors with the same generated target audio sample.

Both experiments were run in multiple iterations with number of epochs ranging from 500 to 1500. This was done to be able to evaluate the performance of the model simultaneously.

5. EVALUATION

5.1 Qualitative Evaluation

A qualitative evaluation has been done by visual inspection and by ear. The waveforms and spectrograms for the "dark" and "bright" experiments can be seen in figures 3 and 4. In the case of estimating the parameters for the "dark" reverb an audio listening inspection indicates an almost identical reverb. For the "bright" reverb it is possible to hear a slightly longer decay time in the high frequency spectrum that might be fixed with further training. By using the calculated loss, and by calculating mean-squared-error (mse) and mean-absolute-error (mae) for target audio y and output audio $\hat{y}$ in equations 15 and 16, we get the values reported in table 4.

$$mse = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{15}$$

$$mae = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \tag{16}$$

For both experiments the models were still converging before termination, but the rate of convergence was slow.

5.2 Quantitative Evaluation

The performance of the model can be evaluated by looking at the training time for the two experiments. The "dark" reverb trained for approximately 20 hours while the "bright" reverb trained for approximately 56 hours. Both tests where run on a 3 Titan X GPU setup with a 2.4 GHz Intel Xeon E5 processor.

6. CONCLUSION AND FUTURE WORK

A neural network model has been created that is able to incorporate VST3 plugins created in JUCE. This incorporation allows for parameter estimation of a custom FDN

reverberator using differentiable audio processing. The parameter estimation model has shown that it is able to converge towards a set of parameters that allows the reverberator to recreate a given input signal. Usable results have been achieved when the reverberation algorithm architecture has been the same for both target and generated audio. But the rate at which the model estimates the parameters is very slow when using the FD gradient estimation.

Two estimation examples have shown the possibilities of the network, but also the current limitations and computational weaknesses. The first experiment gave good initial results with a convergence towards a matching reverb. The second experiment showed considerably more issues with the convergence. A toy model with a simplified reverberation algorithm has shown the ability to quickly estimate parameters utilizing SPSA and a higher epsilon value, but the gap between a single delay line implementation and a 16 delay lines implementation with 16 different filters must be still bridged.

6.1 Future Work

Considering the weaknesses, it is possible to list multiple efforts that should be made in the future to optimize the neural network model and encourage further use. Future work will focus on speed, loss functions, on the parameterization of delay and matrix of the FDN, and on the expansion of the algorithm. Using Finite Differences (FD) as a gradient estimator has a high computational cost. This could be optimized by creating an environment where the SPSA method would prove useful. The convergence of the neural network slows down greatly over time, signaling that we might not have settled on the best hyper-parameters for this problem. Running a hyper-parameter estimation of the neural network is a time-consuming task but should yield better results when moving forward. Further investigation into loss methods might be beneficial for the network. There are many different approaches to utilizing loss functions with regards to audio signals that might warrant

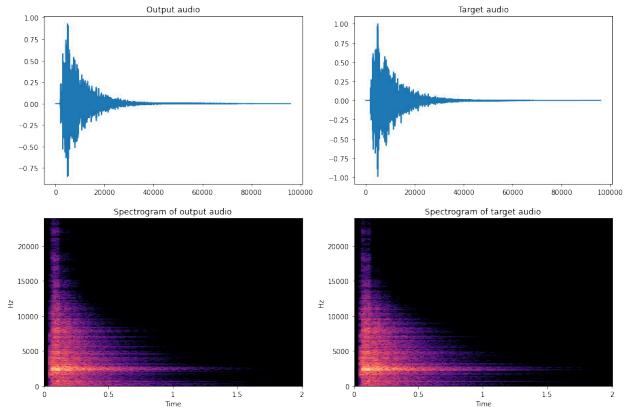


Figure 3: Dark reverb. Upper left figure shows the waveform of the output audio, upper right shows the waveform for the target audio. Lower left shows the spectrogram of the output audio, lower left shows the spectrogram out the target audio.

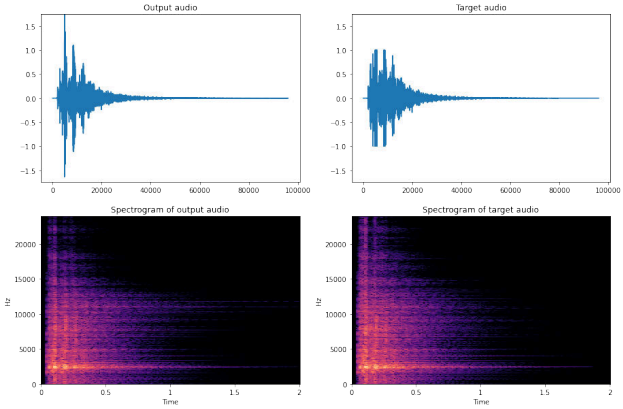


Figure 4: Bright Reverb. Upper left figure shows the waveform of the output audio, upper right shows the waveform for the target audio. Lower left shows the spectrogram of the output audio, lower left shows the spectrogram out the target audio.

an investigation. As the goal of this approach is to emulate different reverberations expansion of the FDN algorithm should be done. By exposing some high-level parameters to the neural network would allow it to be more adaptable to different type of reverberated signals. These high-level parameters should be designed to give the most flexibility while still trying to enforce the rules of losslessness and mutually prime delay lengths. Additionally, delay line modulation would introduce time variance for better physical accuracy. With future work and extension of the project a perceptual evaluation will also be needed to show the viability of technology.

7. REFERENCES

[1] V. Valimaki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[2] J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: differentiable digital signal processing," *CoRR*, vol. abs/2001.04643, 2020. [Online]. Available: https://arxiv.org/abs/2001.04643

[3] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. Int. Conf. Digital Audio Effects (eDAFx-20)*, 2020, pp. 297–303.

[4] M. A. Martínez Ramírez, O. Wang, P. Smaragdis, and N. J. Bryan, "Differentiable signal processing with black-box audio effects," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, June 2021.

[5] Spotify, "Pedalboard," https://github.com/spotify/pedalboard, 2021.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, and et al, "TensorFlow: Large-scale machine learning

on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[7] J.-M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in *Proc. AES Convention*, Feb 1991, preprint 3030.

[8] S. Lee, H.-S. Choi, and K. Lee, "Differentiable artificial reverberation," *arXiv*, 2021.

[9] K. Prawda, S. Willemsen, S. Serafin, and V. Välimäki, "Flexible real-time reverberation synthesis with accurate parameter control," in *23rd International Conference on Digital Audio Effects*, 2020, pp. 16–23.

[10] J. O. Smith, *Physical Audio Signal Processing*. `http://ccrma.stanford.edu/~jos/pasp/`, accessed 16/12/2021, online book, 2010 edition.

[11] A. Wishnick, "Time-varying filters for musical applications." in *DAFx*, 2014, pp. 69–76.

[12] M. C. Fu, "Stochastic gradient estimation," *Handbook of simulation optimization*, pp. 105–147, 2015.

[13] J. C. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins apl technical digest*, vol. 19, no. 4, pp. 482–492, 1998.

[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

STYLEWAVEGAN: STYLE-BASED SYNTHESIS OF DRUM SOUNDS WITH EXTENSIVE CONTROLS USING GENERATIVE ADVERSARIAL NETWORKS

Antoine Lavault
Apeira Technologies
STMS, Sorbonne Université
lavault@ircam.fr

Axel Roebel
STMS, IRCAM, CNRS
Minstère de la Culture
roebel@ircam.fr

Matthieu Voiry
Apeira Technologies
m.voiry@apeira-technologies.fr

ABSTRACT

In this paper we introduce StyleWaveGAN, a style-based drum sound generator that is a variation of StyleGAN, a state-of-the-art image generator [1, 2]. By conditioning StyleWaveGAN on both the type of drum and several audio descriptors, we are able to synthesize waveforms faster than real-time on a GPU directly in CD quality up to a duration of 1.5s while retaining a considerable amount of control over the generation. We also introduce an alternative to the progressive growing of GANs and experimented on the effect of dataset balancing for generative tasks. The experiments are carried out on an augmented subset of a publicly available dataset comprised of different drums and cymbals. We evaluate against two recent drum generators, WaveGAN [3] and NeuroDrum [4], demonstrating significantly improved generation quality (measured with the Frechet Audio Distance) and interesting results with perceptual features.

1. INTRODUCTION

Drum machines are musical devices creating percussion sounds using analogue or digital signal processing [5] [6]. The characteristic sound of this synthesis process contributed to their use in the '80s and their appreciation nowadays. However, these drum machines did not provide an extensive set of controls over the generation.

Following the success of deep learning, several generative processes for percussive sounds have been proposed in the recent years, and two approaches retained our attention. [7] used a GAN for waveform generation with a conditioning on the type of drum, generating 0.3s at 44100kHz. There is also [8], where a GAN was trained to generate STFT of drum sounds while controlling the generator with audio descriptors, allowing them to generate 1s at 16kHz. Both of them used the progressive growing of GANs [9]. Another contribution that does not use GANs is Controllable Raw Audio Synthesis with High-resolution (CRASH) proposed in [10] is a score based generative model that supports a large variety of applications (class conditional synthesis, inpainting, interpolation) but unfortunately suffers from rather long inference times.

In this paper, we build upon the same idea of conditional synthesis using discrete and continuous controls, with time-domain generation like [7] and control by means of perceptual features derived from the AudioCommons project like [4, 8] with a style-based approach (SGAN) [1, 2]. The charactheristics of these networks are summarized in table 1. We expand on the idea of control with perceptual features by means of replacing the trained auxiliary network used in [8, 11] with a differentiable implementation of the feature estimators, increasing the robustness of the feature evaluation. We conduct our experiments on an augmented version of the ENST-Drums [12] dataset, containing kick, snare, toms and hi-hats and comprising about 120k samples amounting to 100 hours of recordings. To evaluate the quality of the model on this dataset, we are using the Fréchet Audio Distance (FAD) [13], in an attempt to obtain a reference-free automatic evaluation of the generated samples. Finally, we explore the ability of the network to use the information from the perceptual features.

All in all, our goal is to create an algorithm for drum sound synthesis suitable for professional music production. In other words, we expect good output quality, real-time generation and relevant controls. The Fréchet Audio Distance (FAD) is used for the quality evaluation, real-time ability is measured through plain generation and the quality of the controls uses the descriptor consistency metric from [4].

Reference	Sample Rate	Duration
WaveGAN [3]	16kHz	1.1s
NeuroDrum [4]	16kHz	1s
DrumGAN [8]	16kHz	1.1s
Drysdale et al. [7]	44.1kHz	0.4s
Ours	**44.1kHz**	**1.5s**

Table 1. Comparison of state of the art neural drum synthesizers

2. MODEL

2.1 Audio-Commons Timbre Models

The Audio Commons project implements a collection of perceptual models that describe high-level timbral char-

Copyright: © 2022 Antoine Lavault et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

acteristics of a sound [14]. These features are specially crafted from the study of popular timbre designations given to a collection of sounds from the Freesound dataset. The perceptual models are built by combining existing low-level features found in the literature [15], which correlate with the chosen timbral designation.

Contrary to [8], we reimplemented those features in order to make them fit directly into the training as differentiable functions. Our motivation behind this comes from the use of an auxiliary network for conditioning in [8]. Constructing a differentiable proxy for these timbral features by training a neural network does not guarantee the correct evaluation of the features to the same degree than implementing the features following the reference implementation. Moreover the direct implementation allows a correct evaluation of signals that have descriptor values outside of the range of values that were available for training the proxy. Our implementation of these descriptors as well as the supplementary material can be found at `https://alavault.github.io/stylewavegan/`

2.2 Generative Adversarial Networks and StyleGAN

Generative Adversarial Networks (GAN) are a family of training procedures in which a generative model (the generator) competes against a discriminative adversary (the discriminator) that learns to distinguish whether a sample is real or fake [16].

Instead of using a vanilla GAN, we are using an evolution called StyleGAN [1, 2]. StyleGAN attempts to mitigate the entangled representation when using noise as latent and input of the generator. The key idea here is to use a *style encoding*, a vector which is obtained through a mapping network and is then used to control (through an affine transform) every layer of a synthesis network.

2.3 Proposed Architecture

Since StyleGAN was originally used for high-quality image generation, we have to modify it for direct waveform generation. In particular, we transform 2D convolution (3×3) into 1D causal convolutions (1×9) [17], the upsampling is done with an averaging filter before each convolution block in the synthesis network, the mapping networks has 4 layers instead of 8 and the loss function is WGAN-LP [18] (see figure 1).

We use the same number of filters, with respect to the depth, as StyleGAN2 [2]. Just like StyleGAN2 , the synthesis network uses input/output skips and the discriminator is a residual network.

In this work we follow [4, 7] using a temporal signal representation. Informal perceptual evaluations performed in the initial phase of this study supported our idea that the temporal representation produces better audio quality than spectral representation : we suppose it is because of the high amount of noise and the importance of the transient in the drum sounds.

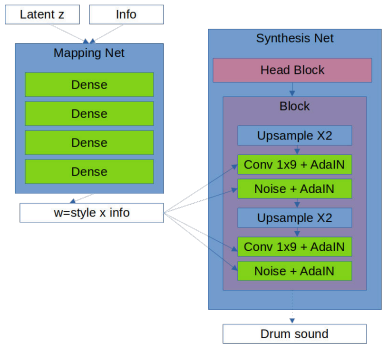


Figure 1. StyleWaveGAN

2.4 Noise Addition Layers and Output Envelopes

We modified the noise addition layers of StyleGAN to make them style-dependant. We also add noise shaping (with a linear fade out) to avoid noisy tails. Having controlled noise addition is useful since some classes need more noise than other to get a good quality synthesis.

This can be summarized in the following equation :

$$y = x + w \cdot n + b \tag{1}$$

where y is the output of the layer, x is the signal input of the layer, w is the transformed style vector, n the shaped noise (the same on every channel) and finally b a bias term.

One of the drawback of having noise addition layers is the lack of control of the decay of said noise. Because of this, the generated sounds have an audible noisy tail which makes them easily identifiable by a human listener. To avoid this pitfall, we added envelopes after the output of the network.

These envelopes were generated using the training dataset, one per type of drum. For each sample of one given type, the final envelope is the filtered mean of the analytical part of the Hilbert transform of these normalized samples. A small fade out is applied to avoid audible clicks at the end of the generated sounds. The Hilbert transform is calculted using the Discrete Fourier Transform on the first 1.5s (65636 samples at 44.1kHz) of each normalized sound of the dataset.

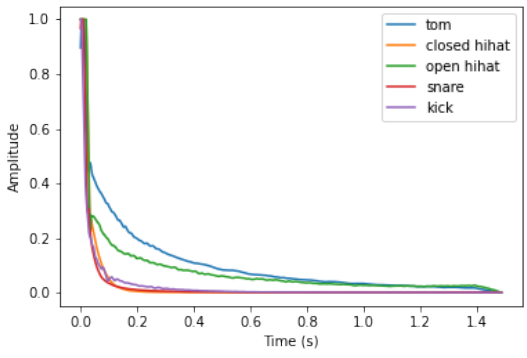


Figure 2. Generated envelopes from the training dataset

The final audio is obtained by multiplying the output of the synthesis network and the matching envelope elementwise. This ensure a quasi-constant energy representation inside the synthesis network. We hypothesize this helps by reducing the dynamic range to generate by the nonlinearities inside the network.

The output time signal of the network y_n is obtained from the output x_n of the network by means of multiplying the envelope signal $e_{n,c}$ for drum class c by means of

$$y_n = x_n e_{n,c} \tag{2}$$

2.5 Controlling the Network

The labels and audio descriptors are fed into an embedding layer which is then concatenated to the latent z (c.f figure 1) and fed to the mapping network. These labels and descriptors are concatenated after the mapping network too.

In our experiments, we are using 5 labels. These labels are added to the network with a one-hot vector. The descriptors, if used, are concatenated after the labels. We expect to have a better disentanglement between the class label or the descriptors during the style encoding by using this method. We use the L1 loss to measure the deviation between the target descriptors and the generated values.

2.6 AutoFade

Progressive growing of GANs has been proposed in [9] and used in [7, 8]. In our experiments, we developed and evaluated a variant of progressive growing, that we denote AutoFade. It is a ResNet architecture with a convolution path and a bypass where a learned parameter is used to fade more or less of one path. Rather than fixing a value like ResNet, we let the network choose the best value as part of the training process, without the need of training it block by block. If x and y represents the two different branches, we have:

$$\sin(\alpha)x + \cos(\alpha)y \tag{3}$$

α is independant of x or y. It makes this structure an intermediate between ResNet and Highway Networks. By using trigonometric function in equation (3), we guarantee the conservation of the standard deviation, if both inputs have equal variance. Similar to [2] we did not find any benefit using progressive growing or AutoFade in the generator. On the other hand using progressive growing in the discriminator did improve the results. The Autofade feature will therefore be evaluated in the following sections, only as part of the discriminator.

All in all, AutoFade is similar to Progressive Growing in the sense where the α parameter changes over time. But contrary to Progressive Growing, the parameter is not forced to increased by hyperparameters but changes with the gradient information. The "growing" is then made dependant of the data and the training iteration.

3. EXPERIMENTAL SETUP

3.1 Dataset

We are using a subset of ENST-Drums [12], comprised of 350 samples of close miking of kicks, snares, toms and hi-hat. Since 350 elements is too low for a data-driven approach, we used an augmentation method similar to [19]. We used `SuperVP` [1] to process the original dataset. The modifications applied to the sounds consist of a gain applied to transient/attack components [20], noise components as well as independent transposition of the signal source and the spectral envelope.

The set of parameters is shown in table 2. The limits have been obtained by means of subjective evaluation of the modified sounds aiming to avoid transformations that can be perceived as unnatural by a human listener. Examples are available in the supplementary material.

As a supplementary metric, the Fréchet Audio Distance between the original dataset and the augmented one is 0.62.

Process	Parameters
Remix attack	0.1, 0.3, 0.6, 1.5, 2, 3
Remix noise	0.6, 1.5, 2, 3
Transposition	0, $\pm$100, $\pm$200
Spectral envelope transposition	0, $\pm$200

Table 2. Augmentation operations and parameters

3.2 Training Procedure

The training procedure is the same as StyleGAN 2 [2], except that we trained the network on 2M samples. With a batch size of 10, it totals to 200k iterations. This takes 7 days without the descriptors and 10 with on a single nVidia 1080GTX.

3.3 Imbalanced Dataset

Balancing datasets is common in classification tasks but to our knowledge, but is quite uncommon for generation tasks. One of such example is [21]. As shown in table 3, our augmented dataset is quite unbalanced, so to obtain a balanced dataset, we use a sampler which takes elements from sub-datasets (one per label) at random according to a uniform distribution. We call it "equal-proportion sampling" : such sampling method does require any downsampling contrary to [21].

3.4 Baseline

The most appropriate candidate to be used as our baseline is DrumGAN [8] and [7]. Unfortunately, these are not reproducible because of missing source code or/and missing or unknown meta parameters. Therefore, we will compare to [4] using the distributed code and a reimplementation of [3], both trained on our augmented dataset.

[1] SuperVP is available free of charge in form of a Max/MSP object at `https://forum.ircam.fr/projects/detail/supervp-for-max/`

Element	Proportion
Kick	3%
Snare	18%
Toms	45%
Closed hi-hat	10%
Open hi-hat	22%

Table 3. Dataset population

Because NeuroDrum [4] works with 16kHz samplerate we adapted our model to use this sample rate for this comparison. We also compared with WaveGAN [3] using our dataset with 44.1kHz. Here we configured both networks to generate 0.3s (@44.1kHz).

3.5 Evaluation

We chose to use the Fréchet Audio Distance (FAD) [13], a reference-free evaluation metric for audio generation algorithms using a VGGish model trained on AudioSet. We compare the embedding of the augmented database to the embedding obtained from 64k samples generated by the evaluated network. In terms of computational cost, we achieve a generation rate of 52drum sounds$/s$ on one 1080GTX with the network in full resolution (1.5s@44.1kHz + descriptors).

Network	FAD
Baseline [4]	25.35
StyleWaveGAN@16kHz	**11.48**

Table 4. FAD comparison to NeuroDrum [4] (lower is better)

Network	FAD
Baseline@44.1kHz [3]	13.08
StyleWaveGAN@44.1kHz (SWG)	**7.75**
SWG + AutoFade (AF)	**6.84**
SWG + Balanced dataset (B)	7.89
SWG + AF + B	7.92

Table 5. FAD on networks without labels (lower is better)

Network	FAD
SWG + labels	6.85
SWG + labels + AF	6.72
SWG + labels + AF + Balanced data (B)	6.65
SWG + labels + AF + B + Envelope	**3.62**

Table 6. FAD on label-conditioned networks (lower is better)

Class	SWG	SWG + AF + B	SWG + AF + B + Env
Kick	8.79	11.71	**3.58**
Snare	7.87	7.53	**4.29**
Tom	8.17	8.09	**6.27**
Closed HH	10.12	6.97	**4.23**
Open HH	8.26	8.91	**4.12**

Table 7. Intra-class FAD for label-conditioned StyleWaveGAN

4. EXPERIMENTAL RESULTS

This section describes the results obtained with StyleWaveGAN on three main configurations. The First unconditioned, the second with conditioning on the labels and finally a third with labels and descriptors.

4.1 Impact of Our Contributions

The first result for unconditioned synthesis we have is that we improved on our baseline in terms of FAD (tables 4 and 5). We can also see from table 5 that using AutoFade in the discriminator helped at getting a better generation in this context.

The results with dataset balancing are mitigated. Without the label conditionning, using it didn't bring any decrease in the FAD : since it makes the training and evaluation dataset different (in proportions), the learned distribution differs, impacting negatively the FAD. This can be seen in table 5. However, it improved the supervised generation, as seen on table 6.

The impact on the intra-class FAD of AutoFade and dataset balancing is shown in table 7. It lowers the FAD generally except for the kick and open hihat. Output envelopes have a very strong impact on the FAD for all drum classes. They reduce the FAD by nearly two for all drum classes besides for the tom.

4.2 Control with Audio Descriptors

We will investigate further on the control of perceptual features. We trained a network on the same dataset, but we made it generate longer audio : 65536 samples, equivalent to 1.48s. Examples are available in the supplementary material.

4.2.1 Brightness

We only focus on one class (snare) and one descriptor (brightness) as a first presentation of the idea. Figure 3 shows the relation between target and synthesized brightness for NeuroDrum and StyleWaveGAN. Results are shown in form of mean values and standard deviation in black dots (StyleWaveGAN) and blue crosses (NeuroDrum). The solid red vertical lines show the limiting values in the training dataset. Finally, the reference target values used for the ordering comparison according to [4, 8] and discussed below are marked with dotted green lines. This figure demonstrates clearly that while the mean value

of the perceptual brightness of a sound produced by NeuroDrum is increasing with the target brightness, it still remains far off the target brightness most of the time. In contrast, the synthesized brightness of StyleWaveGAN is very close to the target value for all values that are present in the training set and even remains somewhat close to the target outside the brightness limits of the training data.

To compare to [4, 8], we are using the ordering criterion used in [4, 8]. It compares pairs of sounds generated with a pair of target values (situated at levels 0.2, 0.5 and 0.8 on a min/max normalized scale), and evaluates whether the ordering of the targets is preserved in the generated features. Like [4], E1 uses extreme points, E2 uses the mid and low values and E3 uses the mid and high values. The very small error in the synthesized feature values generated with StyleWaveGAN results in a consistent ordering for all three criteria. Table 8 reproduces the results for brightness control from table 3 in [8] comparing NeuroDrum and DrumGAN, trained on a different dataset under the column "D1". The results under the columns "D2" are for our network, trained on our augmented dataset. We matched and improved the results from NeuroDrum and DrumGAN in this configuration.

All these results support our hypothesis that replacing a trained feature estimator as in [8, 11] by means of a direct implementation of the feature estimator allows for a significantly improved control consistency of the final network.

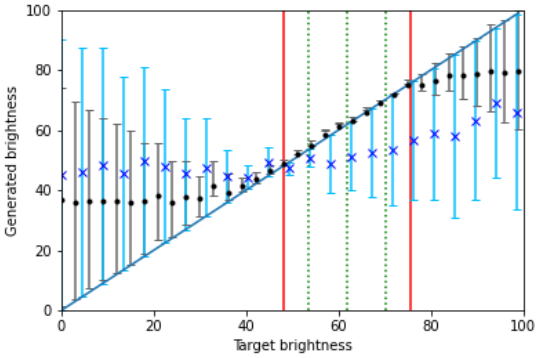


Figure 3. Target brightness vs. Generated brightness (single descriptor). Black dots are for StyleWaveGAN and blue crosses are for NeuroDrum

Features	E1		E2		E3	
Dataset	D1	D2	D1	D2	D1	D2
DrumGAN	0.74	-	0.71	-	0.7	-
NeuroDrum	0.99	0.91	0.99	0.80	0.99	0.68
SWG	-	1.00	-	0.94	-	0.98

Table 8. Ordering accuracy for the feature coherence tests for brightness on samples generated with the baseline NeuroDrum [4] and DrumGAN (from [8]), higher is better

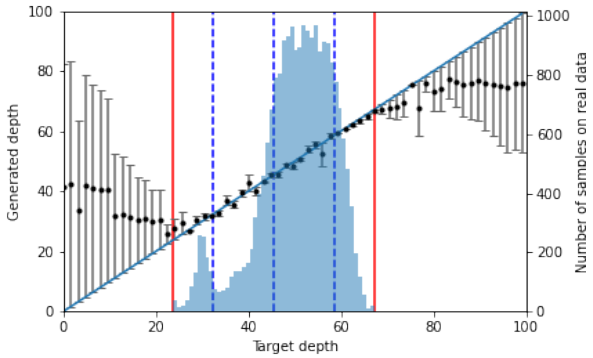


Figure 4. Target depth vs. Generated depth (single descriptor)

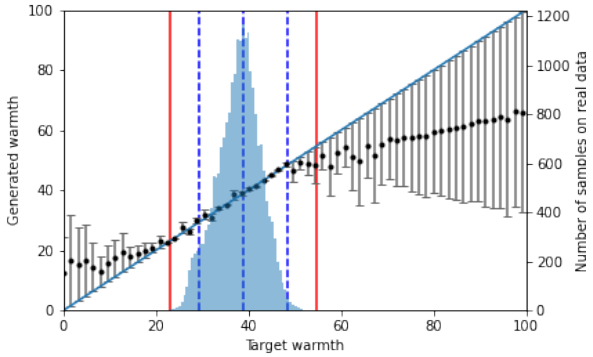


Figure 5. Target warmth vs. Generated warmth (single descriptor)

4.2.2 Other Descriptors

We will discuss here the results for some other descriptors we deem of interest for our task : depth and warmth. Results are shown in table 9 as well as figures 4 and 5. On these figures, an histogram of the dataset values is overlayed in light blue.

Figure 4 shows the results for the depth descriptor. We have a slight worse performance than the brightness descriptor due to some outliers. The same extrapolation property is found here, but slightly less smooth. We can conclude that the depth descriptor is harder to learn for the network. The difference for low depth (< 30, marked by the first blue dashed line) can be explained by the low number of samples to train the network with at this level.

Figure 5 shows the results for the warmth descriptor. The performance is on par with the brightness descriptor except for the region above 80% of the min/max value. This can be explained by a lack of training data is this region as shown on the overlaid histogram.

4.2.3 Multi-dimensional Descriptor Controls

Using three individual networks for controlling the indvidual descriptor is not that interesting for a real world application. In the next step we therefore investigate controlling

Features	E1	E2	E3
Depth	0.99	0.99	0.71
Warmth	1.00	0.86	0.90

Table 9. Ordering accuracy for other features of interest using StyleWaveGAN (higher is better)

the network with a 3 dimensional vector of warmth, depth and brighness descriptors.

When using descriptors simultaneously as part of the control, we can expect conflicts between them as well as dependence to the training data. Since the network is trained on data, it will learn to reproduce similar features as the real data which also means only a part of the combination possibles.

To evaluate the quality of control, we use the same label but change the evaluation method slightly. While we use the same criterion, we generate samples in a way that can create sounds outside of the training dataset. More precisely, we take a set of real features from a batch of the training data and then modify the descriptor to be evaluated to 20, 50 or 80 percent of the min/max value with respect to said descriptor. Results obtained using this method are shown in table 10.

Features	E1	E2	E3
Brightness	1.0	1.0	1.0
Depth	1.0	1.0	0.99
Warmth	0.98	0.59	0.97

Table 10. Ordering accuracy for multiple descriptors using StyleWaveGAN (higher is better)

As shown in table 10, training the descriptors with the proposed differentiable error function produces a network following controls with a precision such that the ordering criterion proposed in [4] and used in [8] is no longer sufficient to evaluate the control precision. In the following we therefore propose a refined evaluation criterion that allows evaluating control precision with more details, taking into account not only ordering but also errors.

In order to achieve this, we will be using the Mean Absolute Error (MAE) between the target values and the output values on three regions based around quantiles of the dataset values :

- F1 : MAE evaluated using only the target descriptor values within the 20th and 50th quantiles
- F2 : MAE evaluated using only the target descriptor values within the 50th and 80th quantiles
- F3 : MAE evaluated using only the target descriptor values within the 20th and 80th quantiles

First, the interest of working with quantiles rather than percentage of the min/max values is that we expect to cover the same amount of values of the dataset each time while avoiding extreme values. The results are shown in table 11. The values given in said table are not percentage or relative to the descriptor values : they are absolute errors. We can also note that these numbers have the same unit as the descriptors.

In table 11, the lines labelled *single* show the results using networks with only one descriptor and the lines labelled *combined* show the results when the descriptor of interest is set but the others are taken from a real sound from the training dataset. Finally, the lines labelled *combined, dataset* show the results when all the descriptors values are taken from the training dataset.

Features	F1	F2	F3
NeuroDrum (brightness)	7.22	10.40	8.81
Brightness (single)	0.83	1.06	0.98
Depth (single)	1.06	1.15	1.10
Warmth (single)	1.15	1.01	1.08
Brightness (combined)	0.97	1.36	1.17
Depth (combined)	1.33	1.50	1.41
Warmth (combined)	1.29	3.31	2.33
Brightness (dataset, combined)	0.75	0.95	0.85
Depth (dataset, combined)	0.99	1.03	1.0
Warmth (dataset, combined)	1.42	1.37	1.39

Table 11. Mean absolute error for several configurations (lower is better)

Since we consider a perfect output follows perfectly the control input, we expect to see a good linear fit on the output. To evaluate this, we will calculate a linear least-square regression on the domain bound by the 20th and 80th quantiles, and use its determination coefficient R^2 as a metric of good linearity. In this case, R^2 is equal to :

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4}$$

where n is the number of samples, y_i is the output value of the i-th measure, $\hat{y_i}$ the corresponding predicted value and $\bar{y}$ the average of the measured values. The results are compiled in table 12. We can also note that we can use the slope from the least-square regression and use it as an ordering criterion.

Features	R^2
NeuroDrum (brightness)	0.03
Brightness (single)	0.75
Depth (single)	0.70
Warmth (single)	0.76
Brightness (combined)	0.47
Depth (combined)	0.67
Warmth (combined)	0.08
Brightness (dataset, combined)	0.72
Depth (dataset, combined)	0.62
Warmth (dataset, combined)	0.45

Table 12. Determination coefficient for several configurations (higher is better)

Apart from a better fit than NeuroDrum, we can see that the R^2 coefficient is generally quite satisfying except for the warmth when used with values outside of the dataset. This illustrated on figure 8, where there is a bend in the output value.

This bend is due to the dataset value distribution, where for high warmth values, the set of values for the other descriptors gets small (a variation of less than 5 points around 50 for brightness and 66 for depth, these values being already quite rare in the dataset). So, when the control inputs gets brightness and depth values that are from the rest of the dataset, the warmth value has to be extrapolated by the network since such combination was not seen during training.

However, this behaviour is not shown when evaluating on control values from the dataset ($0.08 \leftrightarrow 0.45$). For the other descriptors, the linearity remains satisfaying whatever the evaluation method used.

These considerations can be seen on the figures 6 through 8. When iterating on the whole scale (i.e 0 to 100) while setting the other descriptors with values from the training dataset, the output control stays mostly consistent and linear and even allow to generate samples outside the minimum and maximum values of the dataset.

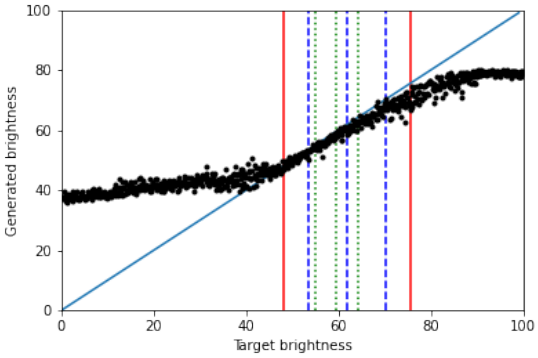


Figure 6. Target brightness vs. Generated brightness with combined descriptors

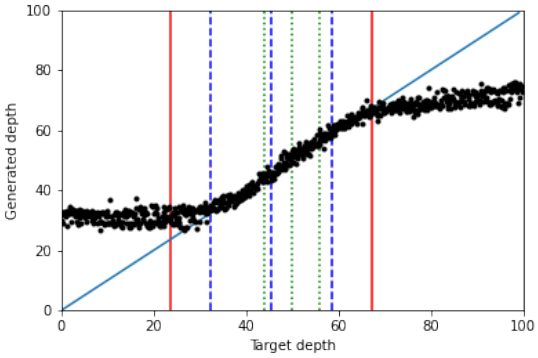


Figure 7. Target depth vs. Generated depth with combined descriptors

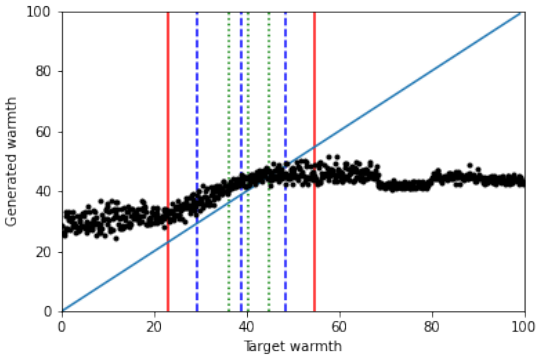


Figure 8. Target warmth vs. Generated warmth with combined descriptors

To conclude, we have justified our method works great almost everywhere in the min/max values of the training dataset and can extrapolate further than the min/max values as well as between unseen combination of descriptors.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented a new method for drum synthesis using StyleWaveGAN, an adaptation of a state of the art image generator. The proposed method has explicit controls on drum type and additional continuous controls for selected perceptive audio features.

We have shown the proposed style-based synthesis achieves a significantly reduced FAD compared to recent DNN based drum synthesis methods [3, 4]. We have proposed a new means for training the feature control by using a differentiable implementation of the AudioCommons features for calculating the feature loss and have demonstrated that this method significantly improves the feature coherence between target and measured features in the synthesized sounds when compared to [4], and argue that the same improvement would hold compared to [8]. We also introduce a way to measure the fidelity of the control with respect to the input. To the best of our knowledge the proposed DNN is the first achieving drum synthesis with 44.1kHz sample rate (for sounds with a duration of 1.5s) with an inference speed more than 50 times faster than real time on a consumer GPU.

In terms of future work we will continue to work on the sound quality and additional controls, notably regarding velocity.

6. REFERENCES

[1] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," 2018. [Online]. Available: http://arxiv.org/abs/1812.04948

[2] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," dec 2019. [Online]. Available: http://arxiv.org/abs/1912.04958

[3] A. Dessein, N. Papadakis, and J. L. Rouas, "Regularized optimal transport and the rot mover's distance," vol. 19, oct 2018, pp. 1–53. [Online]. Available: http://arxiv.org/abs/1610.06447

[4] A. Ramires, P. Chandna, X. Favory, E. Gomez, and X. Serra, "Neural Percussive Synthesis Parameterised by High-Level Timbral Features," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May. Institute of Electrical and Electronics Engineers Inc., nov 2020, pp. 786–790. [Online]. Available: http://arxiv.org/abs/1911.11853

[5] G. Reid, "Practical Snare Drum Synthesis." [Online]. Available: https://www.soundonsound.com/techniques/practical-snare-drum-synthesis

[6] ——, "Practical Cymbal Synthesis." [Online]. Available: https://www.soundonsound.com/techniques/practical-cymbal-synthesis

[7] J. Drysdale, J.; Tomczak, M.; Hockman, "Adversarial Synthesis of Drum S ounds," *Proceedings ofthe 23rd International Conference on Digital Audio Effects (DAFx2020),*, no. September, pp. 24–30, 2020.

[8] J. Nistal, S. Lattner, and G. Richard, "DrumGAN: Synthesis of Drum Sounds With Timbral Feature Conditioning Using Generative Adversarial Networks," 2020. [Online]. Available: http://arxiv.org/abs/2008.12073

[9] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," pp. 1–26, 2017. [Online]. Available: http://arxiv.org/abs/1710.10196

[10] S. Rouard and G. Hadjeres, "CRASH: Raw Audio Score-based Generative Modeling for Controllable High-resolution Drum Sound Synthesis," jun 2021. [Online]. Available: https://arxiv.org/abs/2106.07431v1http://arxiv.org/abs/2106.07431

[11] A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis With Auxiliary Classifier GANs," 2016. [Online]. Available: http://arxiv.org/abs/1610.09585

[12] O. Gillet and G. Richard, "ENST-Drums: An extensive audio-visual database for drum signals processing," *ISMIR 2006 - 7th International Conference on Music Information Retrieval*, pp. 156–159, 2006.

[13] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2019-Septe, pp. 2350–2354, 2019.

[14] A. Pearce, T. Brookes, and R. Mason, "Hierarchical ontology of timbral semantic descriptors," *AudioCommons - Deliverable D5.1*, pp. 1–34, 2016.

[15] G. Peeters, "A Large Set of Audio Features for Sound Description," Tech. Rep. 0, 2004.

[16] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," pp. 1–9, 2014. [Online]. Available: http://arxiv.org/abs/1406.2661

[17] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," sep 2016. [Online]. Available: http://arxiv.org/abs/1609.03499

[18] H. Petzka, A. Fischer, and D. Lukovnicov, "On the regularization of Wasserstein GANs," sep 2017. [Online]. Available: http://arxiv.org/abs/1709.08894

[19] C. Jacques and A. Roebel, "Data Augmentation for Drum Transcription with Convolutional Neural Networks," 2019. [Online]. Available: http://arxiv.org/abs/1903.01416

[20] A. Röbel, "A new approach to transient processing in the phase vocoder," in *Proc. of the 6th Int. Conf. on Digital Audio Effects (DAFx03)*, 2003, pp. 344–349. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01161124

[21] K. Su, X. Liu, and E. Shlizerman, "Audeo: Audio generation for a silent performance video," in *Advances in Neural Information Processing Systems*, vol. 2020-Decem. Neural information processing systems foundation, jun 2020. [Online]. Available: https://arxiv.org/abs/2006.14348v1

A WEB PLATFORM TO EXTRACT AND INVESTIGATE MUSIC GENRE LABELS IN SPOTIFY

Adriano Baratè
Laboratory of Music Informatics (LIM)
Department of Computer Science
University of Milan
adriano.barate@unimi.it

Luca A. Ludovico
Laboratory of Music Informatics (LIM)
Department of Computer Science
University of Milan
luca.ludovico@unimi.it

ABSTRACT

The growing use of music streaming platforms is changing the listening taste and habits of users. While the problem of defining music genres and classifying compositions accordingly has been widely debated in musicology, little attention has been paid to the redefinition of the concept of genre in the light of new distribution models and technological devices. In this paper, we first describe *SpotiGeM*, an interdisciplinary project involving musicology, sociology, and IT experts, aiming to understand the effects of streaming platforms in general, and Spotify in particular, on the concept of genre. After defining the background, we focus on the Web platform implemented and publicly released to support such research. We list the main findings that emerge from use cases and scenarios. We also propose some directions to generalize the project's outcomes, showing that the Web platform can be employed also to investigate other aspects of individual music consumption, artistic processes, and distribution models.

1. INTRODUCTION

The growing use of streaming platforms for listening to music is radically changing the taste, habits, and attitudes of listeners [1]. One aspect, in particular, has been little studied in detail, so far: the definition and consequent redefinition of the concept of music genre. Streaming platforms such as Spotify propose public playlists called, e.g., "Kitchen Jazz Music" or "Classic Rock Drive", shared by thousands of users and having half a million likes. This investigation on genre labels can be enlightening, not only to investigate listening behaviors through genre-based clustering but also to better understand the relationship between recommendation systems and music genres.

The Web platform described in this work has been developed in the context of a multidisciplinary project of the University of Milan titled "I generi musicali nell'era di Spotify – Costruzione sociale, fruizione computazionale e pratiche produttivo-distributive" (in English: "Music genres in the Spotify era – Social construction, computational fruition and productive-distributive practices"). Such a project, for the sake of brevity *SpotiGeM* from now on, [1] combines sociological, musicological, and computer science skills in order to investigate the transformation of the concept of music genre operated by digital streaming services such as Spotify, YouTube, Apple Music, etc. The diffusion of these platforms has generated a reorganization of the strategies of production, distribution, experience, and marketing of music [2–5]. The presence of algorithms that orient user choices [6–8] resulted in forms of categorization able to overcome the traditional notion of music genre, rather promoting aspects such as the mood and the situational contexts in which listening takes place.

According to [9], a *music genre* is a conventional category that identifies some pieces of music as belonging to a shared tradition or set of conventions. Another definition, currently considered normative, is presented in [10]: music genre is a set of musical events, real or possible, whose course is governed by a definite set of socially accepted rules. Genre labels are problematic for several reasons. First, they are often broad terms that are used to describe artists, albums, and pieces that may vary greatly in their characteristics. For instance, The Beatles, AC/DC and Queen are all "rock" bands, and Madonna, Elton John, and Katy Perry are all considered "pop" artists, but their styles, intentions, and audiences are very different. Oddly enough, sometimes there is the opposite problem, namely a proliferation of sub-genres whose lines are blurred. For example, a comprehensive list of "metal" music sub-genres can embrace more than 40 main items and several sub-items. Moreover, genre labels are often decided by record companies with the intent of targeting a particular type of audience or age group. Music streaming platforms, with their recommender systems and mood/situation-oriented playlists, are further complicating this already articulated picture.

Nowadays, a comprehensive study on the definition of music genre and its redefinition due to streaming services is necessarily a multi-disciplinary activity. To this end, *SpotiGeM* aims to achieve an innovative investigation of the encounter between cultural conventions, computational dynamics, and forms of music consumption following the spread of digital-music streaming platforms, a phenomenon of growing interest for both scholars of various disciplines and the music industry.

The present contribution can be framed in the context of

Copyright: © 2022 Adriano Baratè et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[1] *SpotiGeM* stands for **Spoti**fy's **Ge**nres for **M**usic.

SpotiGeM as the first step, in charge of the IT subgroup, necessary to complete the project's tasks. The goal of the Web platform proposed here is to support musicology and sociology experts by providing an easy-to-use interface to query, cross-correlate, and track Spotify data over time. But the scholarly contribution of this initiative goes beyond, constituting an enabling technology to conduct analyses also outside *SpotiGeM*'s scope. First, the Web platform lets Spotify users perform personal explorations of their music library, highlighting some data that are typically hidden and offering aggregated graphical representations. Moreover, data can be compared across different track collections and users, and downloaded in standard formats, thus paving the way for other kinds of quantitative and qualitative investigations.

The goal of this paper is, on one side, to present the Web platform developed in the context of *SpotiGeM* and, on the other side, to unveil the possibilities offered by its use in a more general research context. For the sake of clarity, the contribution does not address the problem of how the concept of music genre is being redefined, rather it aims to support the investigation of such a process in the era of music streaming platforms.

The rest of the paper is organized as follows: Section 2 lists the research questions of the *SpotiGeM* project, Section 3 discusses the most relevant references in the scientific literature, Section 4 presents the developed Web platform, Section 5 reports some preliminary results emerging from the use of such a platform and explains how this approach can answer the research questions of the project, and, finally, Section 6 draws the conclusions.

2. BACKGROUND

SpotiGeM research team has focused on Spotify, [2] a streaming service of increasing importance for both individual consumption and the music industry. Unlike other platforms that host user-generated content, Spotify is a closed system of infrastructures that regulate and organize the musical offer combining digital methods and qualitative research [11].

The research questions of *SpotiGeM* are:

RQ1 How do music streaming platforms reflect and/or reconfigure the musicological definitions of music genres?

RQ2 How does the music industry (e.g., record companies, music publishers, musicians) dialogue with music platforms, thus contributing to the definition of music genres?

RQ3 How do the affordances of digital platforms influence the relationship with the music genre in individual consumption?

As mentioned before, by combining sociology, musicology, and information technology, *SpotiGeM* aims to offer an innovative and original contribution to the study of music listening as a cultural practice increasingly re-mediated by digital technologies.

3. RELATED WORK

The scientific literature on Spotify has mainly focused on three aspects, so far: music distribution, user-customization of the music experience, and user interactions. Concerning the former aspect, researchers investigated how the record industry relates to the platform, discussing economic and artistic strategies, the establishment of new models, and copyright issues [12–15]. As it regards user-customization of music experience, the attention of scholars focused on how Spotify influences listening methods through its recommendation system [16–18]. Finally, the aspects of interactions among users and interactions between the user and the platform, mainly resulting in the creation of playlists, have been explored in [19, 20].

Recently, several studies based on the qualitative and quantitative analysis of data that can be extracted from Spotify are emerging. For instance, as demonstrated by [21], a cluster analysis of musical attributes computed by Spotify algorithms can help understand what makes a song trend.

The relationship between platforms such as Spotify and the definition of music genres has never been studied in detail. Following the multidisciplinary approach of the project, we explored the subject from a sociological, musicological, and technological point of view.

From a sociological perspective, it was observed how the encounter between the practices of musical consumption and the algorithms that guide navigation on the platform fosters the emergence of listening practices defined as "situational", that is, based on the mood or situation in which music experience takes place. This view refers to the concept of music genre as a cultural construction [22].

From a musicological point of view, many scientific works agree on defining the music genre as a discursive and historicized category, whose construction embraces all the actors involved in the processes of production, circulation, and enjoyment [23–25]. With few exceptions [26–28], the scientific literature lacks in approaching the role of platforms as agents and interfaces for music genre definition, concerning both historically established genres, and the constitution of new ones. In this context, the role of Spotify and other platforms can be evaluated as cultural practice in terms of *musicking*, namely a phenomenon that opens up the research to include all the activities and people involved [29].

Finally, concerning the technological side of the project, IT-related literature focuses on the heterogeneous possibilities offered by Spotify APIs [3] for automatic data analysis [30, 31], platform assessment [32], and the release of advanced services built on top of Spotify [33, 34].

4. THE WEB PLATFORM

To support scholars in answering the research questions cited in Section 2, we have designed and released a Web

[2] https://www.spotify.com/

[3] API stands for application programming interface.

platform to make Spotify data easily browsable. Such an interface has been conceived for the goals of the *SpotiGeM* project, but its approach can be generalized and applied to other contexts, too. Concerning the contribution that the solution can give to the (re)definition of genre, this proposal helps Spotify's users gain awareness about genre information, which is not available in the player. As reported in the official Spotify discussion forum, only API calls can retrieve genre information, thus making it hardly accessible to common users.

The Web platform is available at `https://spotigem.lim.di.unimi.it/`. The premise to access the framework is to have a Spotify account, either premium or free, and to be an authenticated user. Already authenticated users can directly access the homepage. If the user is not registered on Spotify or authenticated, these steps can be performed on `https://www.spotify.com/`. The current user can be changed either by logging out from Spotify or using the ad-hoc control in the *SpotiGeM* platform, then logging in as the new user.

The interface is logically structured in 3 sections, graphically rendered as tabs: *User*, *Playlist*, and *Album*. The first area focuses on the metadata of the current user and the list of his/her playlists, the second area on a given playlist and the list of playlist tracks, and the third area on a given album and the list of album tracks. From the technical point of view, user's, artists', tracks', and playlists' data come from Spotify API.

The goal of each section is primarily to let the user view the data in a human-readable way. The entry point is the list of user's playlists (tab *User*, Figure 1). Each playlist can be clicked and explored in detail (tab *Playlist*, Figure 2), and each track can be clicked to move to the corresponding album and to the list of its tracks (tab *Album*, Figure 3).

As another navigation tool, the last two tabs present search fields to input Spotify ids. In this way, any album and public playlist, even those outside the user's lists, can be explored. Ids can be easily retrieved from the URLs appearing in the Spotify Web player.

Spotify computes and returns via API calls some audio features concerning some score aspects (time signature, key, mode), the mood (danceability, valence, energy, tempo), the physical properties (loudness, speechiness, instrumentalness), and the context (liveness, acousticness) of the piece. For a more detailed explanation of the meaning of audio features and the interpretation of their values, please refer to the official documentation. [4] . These track properties can be explored, both in the *Playlist* and in the *Album* tab, by clicking the small arrow on the right (Figure 4).

The platform allows downloading playlist and album data into Comma Separated Values (CSV) format. This feature has been introduced to enable computer-supported analysis and database import.

At the bottom of the page, the *Playlist* and the *Album* tabs also present charts that graphically show the audio features extracted by Spotify. Here information is aggregated, so as

[4] `https://developer.spotify.com/documentation/web-api/reference/`

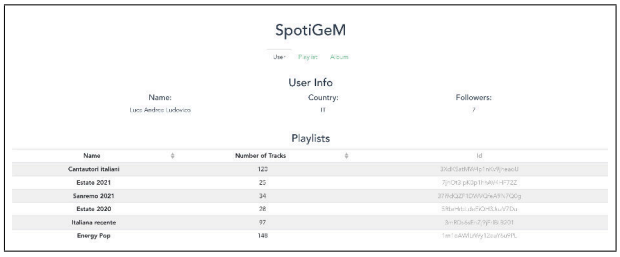


Figure 1. The *User* tab.

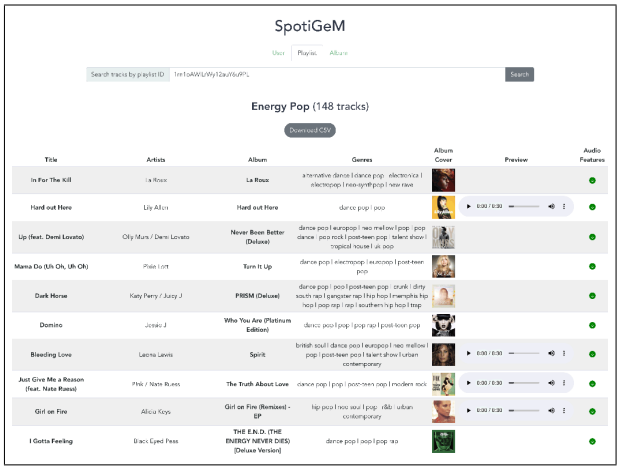


Figure 2. The *Playlist* tab.

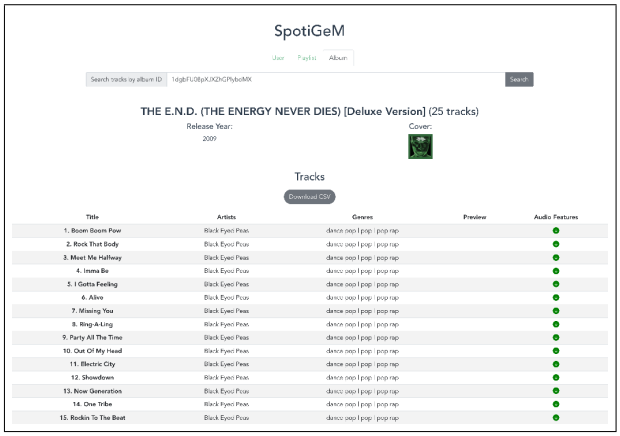


Figure 3. The *Album* tab.

to offer the user a synoptic view of the track collection. For example, the *Keys* explores the distribution of key signatures across the playlist or the album (Figure 5). In our implementation, histograms tend to cluster near values within single bars to improve readability, but most of them can be transformed into dot charts to visualize single values. Finally, plots can be exported as Portable Network Graphics (PNG) files.

4.1 Genre Exploration

In the context of the *SpotiGeM* project, a point of interest is the distribution of genres within and across piece collections, i.e. playlists and albums. For this reason, alongside

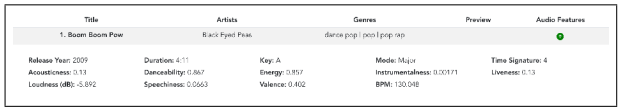


Figure 4. Track features computed by Spotify. The panel is opened and closed by the green arrow in the *Audio features* column.

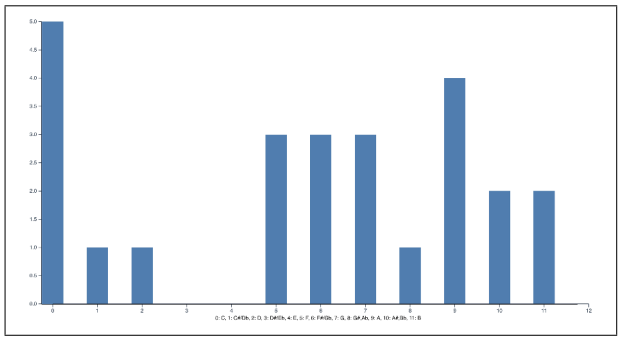


Figure 5. This bar chart shows the distribution of key signatures across a music collection.

the plots dedicated to audio features, there is a chart for music genres with two additional functions:

1. The possibility to exclude underrepresented genres, namely those appearing only once in the whole collection. This option improves the readability of the chart, in general very crowded due to Spotify's approach to genre classification described below;

2. The possibility of weighing the presence of a genre in the playlist in different ways. Any track can potentially present – and, in general, presents – many genre labels. On one side, each occurrence of a given label can increment the total amount of that genre; in this case, "Bohemian Rhapsody", classified as classic rock, glam rock, and rock, adds +1 to the amount of each genre. On the other side, the track is considered to be the bearer of only one genre, and, when many genres occur, they are suitably weighed; in the example above, classic rock, glam rock, and rock add +0.33 to the amount of each genre.

Please note that Spotify does not assign genre information to single tracks. It should be possible to infer such information from the album, and the data structure returned by the API call actually contains such a field but, as a known issue, it is left blank. Consequently, the only available information about the genre has to be retrieved from artists' metadata. Thus, in our platform, tracks inherit genres from artists, in the awareness that such an assumption could introduce ambiguity and inaccuracy.

A paradigmatic example is provided by the track "Cheek to Cheek" performed by Tony Bennett and Lady Gaga. Even if the piece is a well-known jazz standard, in our platform it is labeled by the genres linked to Tony Bennett (adult standards, easy listening, lounge, vocal jazz) plus those linked to Lady Gaga (dance pop, pop, post-teen pop). Thus, vocal jazz, the most plausible genre among those mentioned, is one label out of 7, and the platform user could erroneously infer that "Cheek to Cheek" is about 15% vocal jazz, 15% dance pop, etc.

As a final remark, even artists are not necessarily associated with any genre. For example, the analysis of a dataset made of about 625,000 artists [35] revealed that only 10% had a genre assigned.

These critical issues seem to suggest that Spotify's recommendation system is not fed by genre information, rather it relies on other characteristics, including audio features, track and artist cross-correlation, popularity, collaborative filtering, and Web scraping [36, 37]. Nevertheless, the musicological and sociological investigations of the *SpotiGeM* project do not depend on the actual use of genre-related data in Spotify, but on the complex and evolving relationships between listeners' habits and genres in music streaming services. Under this perspective, the low importance attributed by Spotify to this type of information can constitute a point of reflection in itself.

4.2 Playlist Comparison

Another feature of interest, for both the *SpotiGeM* project and more general studies, is the possibility to perform a numerical and graphical comparison between playlists. Such a tool is accessible from the bottom part of the *Playlist* tab, where the second id can be picked from the shortlist of the other user's playlists or input by hand. The latter option allows comparisons with any other valid playlist, including the thematic collections published by Spotify and the public playlists of other users.

In the framework of *SpotiGeM*, there are mainly two uses of this feature:

1. A *peer mode*, where two different playlists linked by some kind of relationship are studied. An example is comparing a given user's top-hits collection of last summer with the one of 10 years before (Figure 6). Another example is comparing the playlists of favorite rock songs of two users who share some common characteristics in terms of age, education, geographical areas, music abilities, etc.;

2. A *time-machine mode*, where the data of the same playlist are sampled at regular intervals to investigate the evolution in time. Please note that this approach can be applied not only to users' playlists but also to Spotify's trending collections, which are periodically updated.

While the former kind of investigation can be conducted both online (by graphical comparison) and offline (by CSV files), the latter requires downloading data periodically, since in Spotify the playlist id is not renewed when the track composition changes and old versions cannot be retrieved.

5. DISCUSSION

The Web platform presented in Section 4 is a tool that can be applied to heterogeneous research scenarios. It was developed in the framework of *SpotiGeM*, being influenced

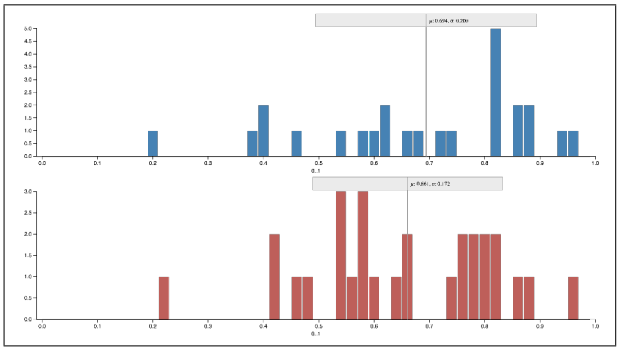


Figure 6. The two bar charts show the evolution of the valence for the author's summer hits collections in 2020 and 2021.

in its functionalities and interface by the sociological and musicological aims of the project, nevertheless, it supports other use patterns (e.g., an alternative exploration of Spotify data by standard users) and can address different research questions by a data-driven approach (e.g., the way Spotify defines audio features such as speechiness and instrumentalness).

The first point to clarify is how to read data since the risk is to confuse the evolution of musical characteristics (for tracks, genres, artists, albums, etc.) with the evolution of users' tastes and listening habits. For example, let us recall the comparison shown in Figure 6, concerning the valence dimension of 2020 and 2021 summer hits chosen by the same user. Changes in the distribution of values could depend on intrinsic track characteristics, but also on the selection operated by the user in two different periods of his/her life. This is the reason why the *SpotiGeM* project is strongly multidisciplinary and the research team includes different competencies and embraces different visions.

After clarifying this aspect, let us go back to the research questions mentioned in Section 2.

RQ1 concerned the way music streaming platforms reflect and/or reconfigure the musicological definitions of music genres. Our investigation focusing on Spotify demonstrated, on one side, the tendency to be exhaustive, defining multiple genres and sub-genres for single artists. For example, Madonna is marked as "pop" and "dance pop", the English band Portishead as "electronica", "laboratorio" and "trip hop", and the famous piano player Liberace as "adult standards", "easy listening" and "kermis". Moreover, even if an exhaustive list has never been officialized, Spotify is supposed to contain an extremely high number of genres. Alongside well-established labels (e.g., "hard rock", "techno", "pop", "jazz", "industrial", "indie pop", "folk", "classical", "heavy metal"), Spotify includes very detailed (e.g., "British garage", "Canadian contemporary r&b", "LGBTQ+ hip hop") and – sometimes – quirk categories (e.g., "animal singing", "neo-singer-songwriter", "wrock"). Spotify's search engine allows queries by genre by typing "genre:name" in the search text box.[5] On the other side, Spotify seems to deliberately ignore the possibility to assign genres to albums and tracks. The Web platform made this aspect clearly emerge, presenting the same information for all the tracks performed by the same artist. For the same reason, tracks with multiple performers have very long lists of genres, coming from the union of all the labels of each artist.

RQ2 focused on the dialogue between the music industry and streaming platforms such as Spotify in the definition of music genres. This relationship is made even more complex by the fact that the biggest major labels, i.e. Sony Music Entertainment, Universal Music Group, and Warner Music Group, partially own and/or have signed deals with Spotify [38–40]. Even if the Web platform does not offer functionalities explicitly conceived to address such an issue, use cases fostered some interesting considerations. For instance, Spotify's playlist titled "This is Buckwheat Zydeco" (id: 37i9dQZF1DZ06evO1KdFST) contains 50 tracks whose genres are "swamp pop" and "zydeco" (*sic*): a curious case of overlap between genre (zydeco) and artist (Buckwheat Zydeco), maybe driven by commercial interests by the music industry. Furthermore, the phenomenon of advertised artists is out in the open. As an example, in 2019 Spotify added a feature that occasionally pops up a full-screen recommendation for a new album of interest for the user, based on a combination of his/her listening taste and human curation. This feature allows artists and their teams to pay to target their fans or acquire new ones potentially interested in their music.

A different phenomenon is the promotion of ad-hoc track collections by firms and companies to directly or indirectly advertise their products. Sponsored playlists, i.e. playlists where commercial products and services are advertised, are widespread, but a more interesting approach is that of branded playlists, where the advertising message is at the core of musical choices. Examples include the Martini music official playlist (id: 7LylxM89goS7AYsIlHUKKO) and the selection of Christmas songs by Coca-Cola (id: 6H2zbgZg8k6i9iq7r7a798). Though not directly traceable to the *music* industry, the intertwining between music and business is an emerging aspect that is worth studying. The target audience is very broad: in April 2020, Spotify was reported to have 286 million monthly users [41]. Tracking the evolution of the most popular playlists published by Spotify – one of the features of the *SpotiGeM* environment – can help understand to what extent such a streaming service can influence the habits and preferences of music listeners, and how advertising sponsorship initiatives affect (and are, in turn, affected) by such a phenomenon.

Finally, RQ3 addressed the way digital platforms influence the relationship with the concept of music genre in the individual music experience. The platform brings out different scenarios, namely the existence of commonly-accepted genres (e.g., "progressive rock", "new wave 80's", etc.), recently defined categories (e.g., "Italian indie", "djent", etc.), and mood-based/situational genres (e.g., "good vibes", "dark & stormy", "Monday motivation", etc.). Spotify and other streaming platforms can actively influence the three scenarios by redefining historical genres, aiming to frame emerging genres, and promoting algorithm-based affective

[5] If a search term consists of more than one word or includes special characters, the label has to be indicated between quotation marks.

pseudo-genres. The Web platform we developed can play a concrete role in such an analysis, providing the possibility to empirically and innovatively evaluate the reconfiguration or generation from scratch of musical categories. The individual user's listening behaviors can be either assessed by themselves or compared with other users.

The results of the *SpotiGeM* project will be produced by scholars and experts in sociology and musicology, after multiple user sessions and focus groups. In this framework, the developed tool will play a fundamental role. The present paper, on one side, is a way to present the Web platform and the general project to the scientific community, and, on the other, aims to encourage further research on the subject.

6. CONCLUSIONS

SpotiGeM is a multidisciplinary project joining musicological, sociological, and digital competencies to analyze the complex and rapidly evolving phenomenon of digital music consumption via streaming platforms. Narrowing the field of investigation to a specific subject, the project aims to stimulate a debate regarding the definition and redefinition of the concept of music genre as a consequence of new models of music experience.

This paper has introduced the technological tool designed and implemented in the framework of *SpotiGeM* to support the musicological and sociological investigation. This is the first step of the initiative, namely an enabling technology to let experts from the mentioned domains conduct their research. Anyway, the valence of the Web platform, publicly and freely available, is more general. It can also be employed, e.g., as an alternative viewer for users' playlists, or as a way to unveil some hidden Spotify's characteristics, such as the extraction of track-by-track audio features.

Concerning the future development of the Web platform, we are planning to extend some of its functionalities and introduce new ones. Since *SpotiGeM* was a user-centered study, only playlist comparison was relevant for the goals of the project. Nevertheless, implementing album comparison could be useful to deepen the evolution of an artist (e.g., early vs. late albums of the Beatles) or match the musical production of contemporary performers (e.g., the Sex Pistols vs. the Clash).

Another future direction is the characterization of users via the creation of a profile page based on Spotify data. So far, the interface has been presenting the information from API calls with no other form of aggregation than playlists and albums. Conversely, synoptic tables and charts fed by a full load of Spotify data, possibly tracked over time, could effectively draw a comprehensive picture of users' listening tastes and habits.

Finally, we are studying the possibilities to generalize the described approach. We would like to turn a user-based application into a large-scale investigation tool, relying not only on selected users and focus groups but also on crowd sourcing and data donation from users. The final goal is the exploration of Spotify's entire genre space, the universe of mood-based and situational playlists, and their mutual interactions.

Acknowledgments

This research was funded by the University of Milan in the context of the SEED 2019 grant (Bando SEED – Linea 3 del Piano di Sostegno alla Ricerca), an extraordinary call for interdepartmental projects published in May 2019.

The authors wish to acknowledge Alessandro Gandini (University of Milan, Department of Social and Political Sciences, principal investigator of the *SpotiGeM* project), and Maurizio Corbella (University of Milan, Department of Cultural and Environmental Heritage, project member) for their valuable contributions and help.

7. REFERENCES

[1] P. Wikström, *The music industry: Music in the cloud*. Hoboken, New Jersey, USA: John Wiley & Sons, 2020.

[2] K. Swanson, "A case study on Spotify: Exploring perceptions of the music streaming service," *Journal of the Music and Entertainment Industry Educators Association*, vol. 13, no. 1, pp. 207–230, 2013.

[3] M. Hogan, "Upstream effects of the streaming revolution: A look into the law and economics of a Spotify-dominated music industry," *Colorado Technology Law Journal*, vol. 14, no. 1, pp. 131–152, 2015.

[4] A. Coffey, *The impact that music streaming services such as Spotify, Tidal and Apple Music have had on consumers, artists and the music industry itself*. Dublin, Ireland: University of Dublin, 2016.

[5] G. Bhoot, *Music Industry Sales: How streaming services such as Spotify, Apple Music and TIDAL affect album sales*. San Luis Obispo, California, USA: California Polytechnic State University, 2017.

[6] A. Anderson, L. Maystre, I. Anderson, R. Mehrotra, and M. Lalmas, "Algorithmic effects on the diversity of consumption on Spotify," in *Proceedings of The Web Conference 2020*, 2020, pp. 2155–2165.

[7] A. Werner, "Organizing music, organizing gender: algorithmic culture and Spotify recommendations," *Popular Communication*, vol. 18, no. 1, pp. 78–90, 2020.

[8] A.-B. Gran, P. Booth, and T. Bucher, "To be or not to be algorithm aware: a question of a new digital divide?" *Information, Communication & Society*, vol. 24, no. 12, pp. 1779–1796, 2021.

[9] J. Samson, "Genre," in *Grove Music Online*, 2001.

[10] F. Fabbri, *A theory of musical genres. Two applications*. Torino, Italy: International Association for the Study Of Popular Music, 1982.

[11] M. Eriksson, R. Fleischer, A. Johansson, P. Snickars, and P. Vonderau, *Spotify teardown: Inside the black box of streaming music*. Cambridge, Massachusetts, USA: Mit Press, 2019.

[12] R. McLean, P. G. Oliver, and D. W. Wainwright, "The myths of empowerment through information communication technologies: An exploration of the music industries and fan bases," *Management Decision*, vol. 48, no. 9, pp. 1365–1377, 2010.

[13] L. Marshall, "'Let's keep music special. F—Spotify': on-demand streaming and the controversy over artist royalties," *Creative Industries Journal*, vol. 8, no. 2, pp. 177–189, 2015.

[14] J. W. Morris and D. Powers, "Control, curation and musical experience in streaming music services," *Creative Industries Journal*, vol. 8, no. 2, pp. 106–122, 2015.

[15] R. Fleischer, "Universal spotification? the shifting meanings of "Spotify" as a model for the media industries," *Popular Communication*, vol. 19, no. 1, pp. 14–25, 2021.

[16] M. Pichl, E. Zangerle, and G. Specht, "Combining Spotify and Twitter data for generating a recent and public dataset for music recommendation." in *Grundlagen von Datenbanken*, 2014, pp. 35–40.

[17] K. Jacobson, V. Murali, E. Newett, B. Whitman, and R. Yon, "Music personalization at Spotify," in *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016, pp. 373–373.

[18] R. Prey, "Locating power in platformization: Music streaming playlists and curatorial power," *Social Media + Society*, vol. 6, no. 2, p. 2056305120933291, 2020.

[19] M. Pichl, E. Zangerle, and G. Specht, "Understanding user-curated playlists on Spotify: A machine learning approach," *International Journal of Multimedia Data Engineering and Management (IJMDEM)*, vol. 8, no. 4, pp. 44–59, 2017.

[20] B. Zhang, G. Kreitz, M. Isaksson, J. Ubillos, G. Urdaneta, J. A. Pouwelse, and D. Epema, "Understanding user behavior in Spotify," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 220–224.

[21] Z. Al-Beitawi, M. Salehan, and S. Zhang, "What makes a song trend? cluster analysis of musical attributes for Spotify top trending songs," *Journal of Marketing Development and Competitiveness*, vol. 14, no. 3, pp. 79–91, 2020.

[22] M. Airoldi, D. Beraldo, and A. Gandini, "Follow the algorithm: An exploratory investigation of music on YouTube," *Poetics*, vol. 57, pp. 1–13, 2016.

[23] F. Holt, *Genre in popular music*. Chicago, Illinois, USA: University of Chicago Press, 2009.

[24] F. Fabbri, "How genres are born, change, die: Conventions, communities and diachronic processes," in *Critical Musicological Reflections*, S. Hawkins, Ed. London, UK: Routledge, 2012, pp. 179–191.

[25] J. Ehmann, "Categorizing sound: Genre and twentieth-century popular music. by david brackett. oakland, ca: University of california press, 2016. 368 pp. isbn 9780520248717," *Popular Music*, vol. 36, no. 2, pp. 326–328, 2017.

[26] K. Barr, "Theorizing music streaming: Preliminary investigations," *Scottish Music Review*, vol. 3, no. 2, pp. 1–20, 2013.

[27] P. Burkart, "Music in the cloud and the digital sublime," *Popular Music and Society*, vol. 37, no. 4, pp. 393–407, 2014.

[28] A. N. Hagen, "The playlist experience: Personal playlists in music streaming services," *Popular Music and Society*, vol. 38, no. 5, pp. 625–645, 2015.

[29] C. Small, *Musicking: The meanings of performing and listening*. Middletown, Connecticut, USA: Wesleyan University Press, 1998.

[30] T. South, "Network analysis of the Spotify artist collaboration graph," *Australian Mathematical Sciences Institute*, pp. 1–12, 2018.

[31] R. de Fleurian and M. T. Pearce, "The relationship between valence and chills in music: A corpus analysis," *i-Perception*, vol. 12, no. 4, p. 20416695211024680, 2021.

[32] R. Panda, H. Redinho, C. Gonçalves, R. Malheiro, and R. P. Paiva, "How does the Spotify API compare to the music emotion recognition state-of-the-art?" in *Proceedings of the 18th Sound and Music Computing Conference (SMC 2021)*. Axea sas/SMC Network, 2021, pp. 238–245.

[33] A. Germain and J. Chakareski, "Spotify me: Facebook-assisted automatic playlist generation," in *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2013, pp. 025–028.

[34] M. Pichl, E. Zangerle, and G. Specht, "#nowplaying on #Spotify: leveraging Spotify information on Twitter for artist recommendations," in *International Conference on Web Engineering*. Springer, 2015, pp. 163–174.

[35] T. South, M. Roughan, and L. Mitchell, "Popularity and centrality in Spotify networks: critical transitions in eigenvector centrality," *Journal of Complex Networks*, vol. 8, no. 6, 03 2021.

[36] J. Cohn, *The burden of choice: Recommendations, subversion, and algorithmic culture*. New Brunswick, New Jersey, USA: Rutgers University Press, 2019.

[37] K. E. Goldschmitt and N. Seaver, *Shaping the Stream: Techniques and Troubles of Algorithmic Recommendation*, ser. Cambridge Companions to Music. Cambridge, UK: Cambridge University Press, 2019, p. 63–81.

[38] T. Ingham, "Who are the two 'major labels' that have signed new deals with Spotify? clue: not universal or warner," *Music Business Worldwide*, 2019. [Online]. Available: https://www.musicbusinessworldwide.com/who-are-the-two-major-labels-that-have-signed-new-deals-with-spotify-clue-not-universal-or-warner/

[39] ——, "Who really owns Spotify?" *Rolling Stone*, 2020. [Online]. Available: https://www.rollingstone.com/pro/news/who-really-owns-spotify-955388/

[40] A. Zarczynski, "Spotify's new universal music group deal drives deeper artist-fan engagement," *Forbes*, 2020. [Online]. Available: https://www.forbes.com/sites/andreazarczynski/2020/07/22/spotifys-new-universal-music-group-deal-drives-deeper-artist-fan-engagement/

[41] S. Dredge, "Despite Covid-19 Spotify grew its subscribers to 130m in q1 2020," *Music Ally*, April 2020. [Online]. Available: https://musically.com/2020/04/29/spotify-financials-q1-2020-covid-19/

SEMI-SUPERVISED CONVOLUTIVE NMF FOR AUTOMATIC PIANO TRANSCRIPTION

Haoran Wu
INSA Rennes, France
haoran.wu@insa-rennes.fr

Axel Marmoret
Univ Rennes, Inria, CNRS,
IRISA, France
axel.marmoret@inria.fr

Jérémy E. Cohen
Univ Lyon, INSA-Lyon, UCBL,
UJM-Saint Etienne, CNRS, Inserm,
CREATIS UMR5220, U1206, France
jeremy.cohen@cnrs.fr

ABSTRACT

Automatic Music Transcription, which consists in transforming an audio recording of a musical performance into symbolic format, remains a difficult Music Information Retrieval task. In this work, which focuses on piano transcription, we propose a semi-supervised approach using low-rank matrix factorization techniques, in particular Convolutive Nonnegative Matrix Factorization. In the semi-supervised setting, only a single recording of each individual notes is required. We show on the MAPS dataset that the proposed semi-supervised CNMF method performs better than state-of-the-art low-rank factorization techniques and a little worse than supervised deep learning state-of-the-art methods, while however suffering from generalization issues.

1. INTRODUCTION

Automatic Music Transcription (AMT) is the task of transforming music recordings into symbolic format, such as scores or MIDI. It is a fundamental musical skill to acquire, taught from early age up to professional level in music schools and, given enough training, humans can be extremely accurate at transcription. Automatic music transcription aims at accelerating and improving time-consuming manual transcription and has applications in music tutoring and rehearsing, musicology analysis or in other music information retrieval tasks [1].

However, while audio generation from MIDI is rather mature, its counterpart AMT is still a very challenging task, even in scenarios involving a single multipitch instrument like a piano, which is our case study. As reported in the 2018 survey by Benetos *et. al.* [1], there are mainly two families of methods to perform AMT: 1) Methods based on low-rank factorizations of spectrograms, and in particular Nonnegative Matrix Factorization (NMF). These methods are mostly unsupervised [2–4]. 2) Deep Neural Networks (DNN) which are heavily supervised. They require registered symbolic-audio training data in a large amount, which can be hard to acquire [5–9].

Copyright: © 2022 Haoran Wu et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

A recent outbreak in the task of piano transcription (as well as other related tasks) is due to the release of the MAESTRO dataset [7], a large dataset of tightly matched MIDI and audio piano recordings of professional quality which improved the training quality of deep learning techniques. However, the supervised methods require extensive amounts of training data which may not be available for most instruments. The quality of the MAESTRO dataset comes from the existence of the Yamaha Disklavier™, which enables co-recording of audio and MIDI. This high level technology does not exist for most instruments, and building large training dataset for most polyphonic instruments would be extremely challenging on the practical side.

In contrast, since unsupervised factorization-based approaches do not require training data, they obviously solve the data frugality and generalization problems at the cost of being far less accurate than deep supervised approaches.

The goal of this paper is two-fold. On a first hand, leveraging training data available only in limited quantity. On another hand, deploying a variant of NMF, coined Convolutive NMF, in the context of transcription, to improve the transcription performance with respect to NMF. The most closely related work is surely the Attack Decay model [3], which also performs semi-supervision, and proposes a model reminiscent of CNMF. The major differences between the proposed CNMF framework and this work of Cheng *et. al.* are discussed in Section 2.2. Moreover, in Section 4, we show that the performance of the proposed approach are generally much higher and can reach the performance levels observed with Deep Learning at the cost of poor generalization properties. In [4], authors also consider CNMF for piano transcription but CNMF is not the main focus of their work.

This paper is organized as follows: in Section 2, we review the basics of NMF and CNMF for transcription. In Section 3, semi-supervised CNMF is introduced. In Section 4 we show experimental results on MAPS and MAESTRO. Section 5 is devoted to discussions and perspectives.

Notations: Matrices and higher-order arrays are denoted by capital letters, T_{ijk} is the element (i, j, k) in the three-way array T. To denote slices, we use semicolons, so that $T_{i::}$ denotes for instance the slice of all elements of T on row i. Finally, we denote $T_{[a:b]jk}$ elements (i, j, k) with $i \in [a, b]$.

2. CNMF FOR TRANSCRIPTION

2.1 NMF and CNMF Formalisms

Given an element-wise nonnegative matrix $M \in \mathbb{R}_+^{n\times m}$ indexed as M_{ft} with $f \in [1,n],\ t \in [1,m]$, Nonnegative Matrix Factorization (NMF) is a low-rank approximation technique that summarizes M as a sum of rank-one parts, such that

$$M_{ft} = \sum_{q=1}^{r} W_{fq} H_{qt} \tag{1}$$

where $r \leq \min(n,m)$ is a user-defined parameter relating to the number of patterns underlying M, see Figure 1. In practice, when M is an amplitude spectrogram, such as in this work, NMF is computed approximately and boils down to solving a bi-level constrained optimization problem

$$\underset{W\in\mathbb{R}_+^{n\times r}, H\in\mathbb{R}_+^{r\times m}}{\operatorname{argmin}} D_{KL}(M, WH) \tag{2}$$

where $D_{KL}(M,WH)$ is the element-wise Kullback-Leibler divergence between matrix M and its nonnegative low-rank approximation $WH = \sum_{q=1}^{r} W_{:q}H_{q:}$. In AMT, parameter r often relates to the number of notes expected in the recording, and therefore is generally set to (sometimes a multiple of) $r = 88$ for piano recordings [2].

Furthermore, factor matrices W and H are respectively related to pitch and time activation. More specifically, each column of W is expected to contain a spectral template characteristic of a single pitch on the instrument used in the recording, while each corresponding row in H is expected to provide the activation of that note in the recording [10], see Figure 2.

An immediate critic about applying NMF to AMT is that reducing a note to a single frequency template, even tailored for a given instrument, is too restrictive. In practice, frequency templates should evolve with both amplitude and time. While explicit amplitude dependence would break the principle of low-rank approximation underlying NMF, it is possible to extend NMF to include a time-dependence on the templates, which yields Convolutive NMF [12]:

$$\underset{W\in\mathbb{R}_+^{n\times\tau\times r}, H\in\mathbb{R}_+^{r\times m}}{\operatorname{argmin}} D_{KL}(M, \sum_{q=1}^{r} W_{::q} * H_{q:}) \tag{3}$$

where $[W_{::q} * H_{q:}]_{:t} = \sum_{i=0}^{\tau-1} W_{:iq}H_{q(t-i)}$ is a discrete convolution and $q \in [1,r]$, see Figure 1 for an illustration. By convention, we set $H_{q(t-i)} = 0$ whenever $t-i \leq 0$. Integer τ is again a user-defined hyperparameter that dictates the size of the convolution window. To provide a different perspective, the element-wise noiseless CNMF also writes

$$M_{ft} = \sum_{q=1}^{r}\sum_{i=0}^{\tau-1} W_{fiq}H_{q(t-i)}\ . \tag{4}$$

In a nutshell, CNMF enriches NMF by allowing each note to have a full STFT matrix $W_{::q}$ as a frequency template instead of a single column. Therefore, it may also captures time-dependent events such as echoes or non-uniform partials attenuation. It can also be interpreted as a constrained NMF with large rank $r \times \tau$ where each note is represented by τ templates, and the corresponding τ rows in H are constrained to be equal up to a shift. Other works have also considered enriching NMF with several templates per note albeit not using convolution, typically by fusing rows of the estimated H matrix a posteriori [13–15].

2.2 Comparing the Attack Decay Model With CNMF

A reader familiar with the work of Cheng *et. al.* [3] will notice that our work is similar in several aspects with their proposed Attack Decay (AD) framework for music transcription, but let us properly compare the models. After some rewriting of the original AD (see additional material [1]), AD decomposes the data M into two terms

$$M_{ft} = \sum_{q=1}^{r} \underbrace{\sum_{i=0}^{2\tau} \left(\tilde{W}_{fq}^{\text{attack}} P_{-i}\right) H_{q(t-i)}}_{\text{one note attack}} \tag{5}$$

$$+ \sum_{q=1}^{r} \underbrace{\sum_{i=\tau}^{t+\tau-1} \left(\tilde{W}_{fq}^{\text{decay}} e^{-\alpha_q(i-\tau)}\right) H_{q(t-i)}}_{\text{one note decay}}\ . \tag{6}$$

It thus appears that the attack term is a CNMF with rank-one templates $W_{fiq}^{(\text{CNMF})} = W_{fq}^{\text{a}} P_i$ which is therefore less general than the CNMF model. The decay term is also a CNMF with rank-one templates.

With some further manipulations, one can see that it is possible to entirely recast the AD model as a CNMF model with rank-two templates, which may explain the performance gap between the two models observed in Section 4. Indeed in the semi-supervised setting, we seem to have enough data to learn unconstrained templates W^{train}, and the Attack-Decay structure on the templates may not be beneficial.

3. TEMPLATE LEARNING AND CNMF

3.1 Challenges in Unsupervised CNMF

In the context of music transcription, it is rarely discussed why NMF performs extremely well on simple dataset, but rather poorly on more complex ones. Saying that NMF, or CNMF, is a part-based representation with no destructive interferences between components does not explain this behavior. In fact, supposing the data indeed is generated reasonably well with a "ground-truth" NMF $M = AB$ for some true frequency templates $A \in \mathbb{R}_+^{n\times r}$ and activations $B \in \mathbb{R}_+^{r\times m}$, we need to ensure that computing an exact NMF $M = WH$ will indeed yield $A = W$ and $B = H$. In other words, the data M must admit a unique NMF.

Theoretically speaking, it is known that NMF will only enjoy this uniqueness property in particular cases, such as when sources are sufficiently scattered or when the data is very sparse [16, 17]. While this may hold for simple songs where notes do not overlap a lot, in the general case one should **not** expect that W and H behave as expected without restricting the set of solutions. Even worse, CNMF

[1] https://github.com/cohenjer/TransSSCNMF

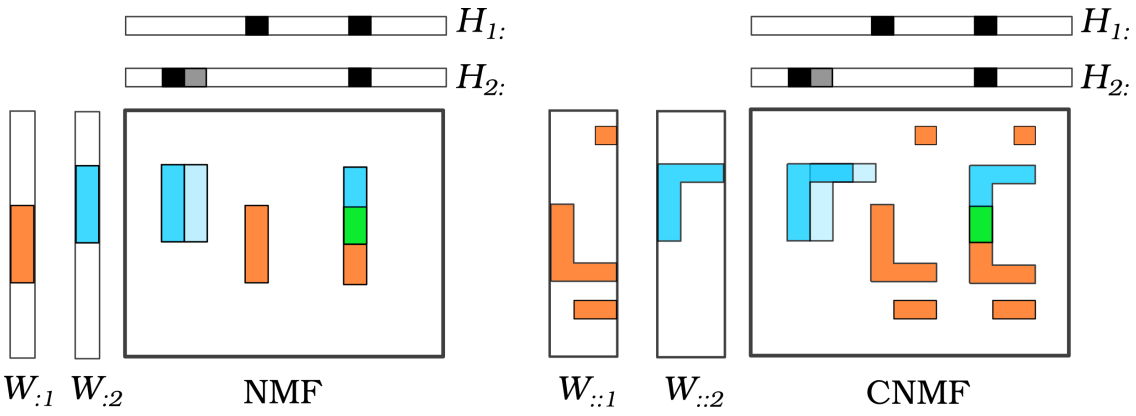


Figure 1. A visual comparison of NMF (left) and CNMF (right). CNMF allows to model complex time dependance while maintaining the number of templates low.

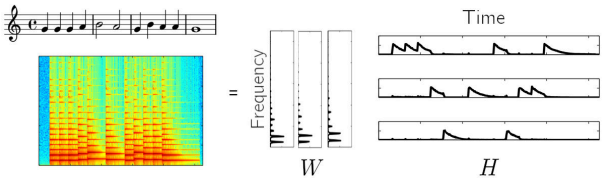


Figure 2. A toy example of transcription using NMF (adapted from [11]).

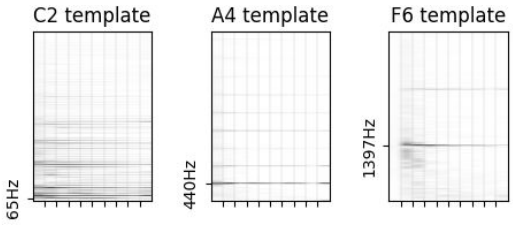


Figure 3. Three trained templates from the AkPnCGdD synthetic piano in MAPS, using $\tau = 10$ convolution size. Templates $W_{::q}$ have been square rooted to better highlight higher frequencies.

being a generalized NMF model, it is bound to have even weaker uniqueness properties than NMF (but nothing is known on CNMF identifiability to the best of our knowledge). Blind CNMF has been used with additional sparsity constraints for drums transcription, but dealing with drums typically yields much sparser and lower-rank data than pitched audio due to the temporal localization of percussive sounds.

Therefore, in general, unsupervised CNMF is not regularized enough to perform transcription. While some works focus on further regularization of NMF [18], we instead turn towards semi-supervision.

3.2 Learning Note-Wise Templates

Our working hypothesis is that audio recordings of isolated pitches are available, similarly to what is used for virtual instruments, except that we only make use of one template per note. Each recording is processed as the module of its complex STFT, denoted $V_{::q} \in \mathbb{R}_+^{n \times m_q}$ where m_q is the number of STFT frames for that recording. For a regular piano one needs 88 such templates. Apart from pitch knowledge, no registered MIDI information is required.

The goal of the learning phase here is to estimate $W_{::q}$ for each q using each individual recording $V_{::q}$. We propose to compute an approximate rank-one CNMF of each $V_{::q}$ to estimate $W_{::q}$ and h_q^{train}, the latter being discarded after the training phase. From a theoretical perspective, rank-one CNMF is a constrained version of NMF of rank τ, furthermore computed on a very simple dataset. Therefore it fulfills the qualitative NMF uniqueness criteria discussed above, and we expect the recovered W to contain adequate note frequency templates.

Practically, we solve for each $q \in [1, r]$ the following optimization problem

$$W_{::q}^{\text{train}}, h_{q:}^{\text{train}} \in \underset{W \in \mathbb{R}_+^{n \times \tau \times 1},\, h \in \mathbb{R}_+^{1 \times m}}{\operatorname{argmin}} D_{KL}(V_q, W * h) \quad (7)$$

using a recently proposed multiplicative algorithm [19] which alternates between W and h updates while preserving nonnegativity and ensuring cost decrease.

In spite of the rank-one approximation and the simple data, the optimization problem still proves challenging with many local minima. Therefore initialization plays an important role in the learning phase. Because it is reasonable to look for $W_{::q}$ in the $V_{::q}$ data itself, we set

$$W_{::q}^{\text{init}} = V_{:[t^*:t^*+\tau-1]q} \text{ and } t^* = \underset{t \leq m_q}{\operatorname{argmax}} \|V_{:[t:t+\tau-1]q}\|_1 \quad (8)$$

which amounts to finding the τ consecutive columns with most energy for initialization. Then we fill $h_{q:}^{\text{init}}$ with zeros and place a one at t^*. Note that this initialization procedure mimics a recently proposed algorithm for separable CNMF [2] [20] but is less computationally intensive. A total of 500 outer iterations are performed to learn a single note template.

Once the training phase is over, for a single multipitch

[2] Separable CNMF is a computationally simpler variant of CNMF which looks for all matrices $W_{::q}$ in the data itself.

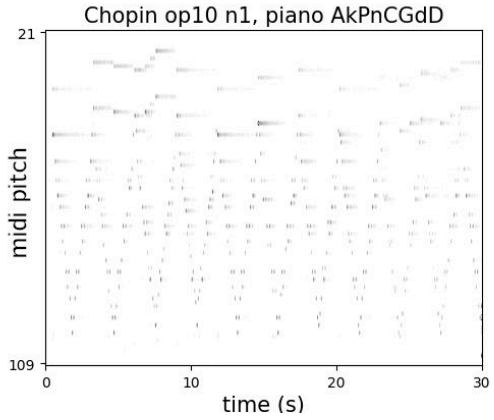


Figure 4. An example of H^{test} computed through rank-one CNMF.

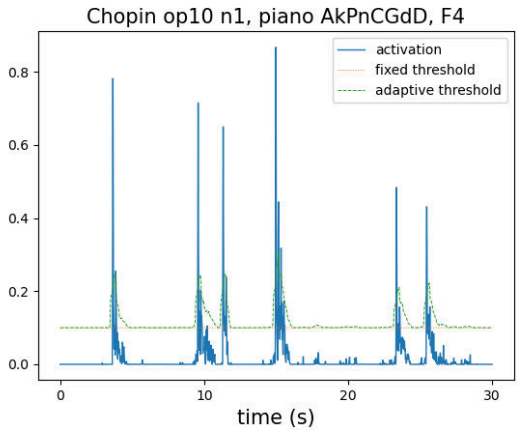


Figure 5. The CNMF activation and adaptive peak-picking method shown for note F4 using a song from MAPS.

instrument, we have at our disposal the whole dictionary W^{train}, see Figure 3.

3.3 CNMF Transcription With Templates

Testing in the semi-supervised framework only consists of computing the time activations H for a given music excerpt M to transcribe, since W has been pre-trained. This makes the transcription task much easier since the problem

$$H^{\text{test}} \in \underset{H \in \mathbb{R}_+^{r \times m}}{\operatorname{argmin}} D_{KL}(M, \sum_{q=1}^{r} W_{::q}^{\text{train}} * H_{q:}) \qquad (9)$$

is convex and therefore can be solved up to arbitrary precision with the algorithm proposed in [19]. In practice 100 iterations are used, which is generally enough to reach convergence. Initialization was carried out using a few iterations of NMF with W fixed as the first column of each trained template $W_{::q}^{\text{train}}$. An example output H^{test} is provided in Figure 4.

3.4 Post-Processing of Activations

The post-processing of H^{test} that produces a MIDI file matters a lot. Hopefully, prior works have already proposed quite efficient post-processing using an adaptive threshold [3]. We essentially use the same technique but simplified.

In short, activation values in each row of H^{test}, averaged over several consecutive frames, are added to a user-defined threshold δ, defining an adaptative threshold. An onset is detected at the position where the signal is above this adaptive threshold, see Figure 5 for an illustration. Formally, an onset is detected at frame t for note q when

$$h_{qt} > \frac{1}{21} \sum_{j=-10}^{10} h_{q(t+j)} + \delta, \qquad (10)$$

using zero-padding when necessary. The activations are typically very sparse, so we generally did not observe spurious double peaks using the adaptive threshold contrarily to what was observed in [3].

4. EXPERIMENTS ON MAPS AND MAESTRO

Although the proposed semi-supervised CNMF framework works in principle for transcribing any multipitch instrument, we only evaluate the performance for piano transcription as a proof on concept. Among the few existing open piano recordings dataset with registered audio and MIDI, in Section 4.2 we focused especially on MAPS [21] which has several kinds of individual notes recordings for several pianos, both virtual and acoustic. We also used MAESTRO [7] to evaluate generalization performance in Section 4.3. In our tests, we only considered the first 30 seconds of each song, as in [3]. Results are discussed in Section 5.

4.1 Experimental Setup

Let us briefly state the various experimental parameters required to reproduce the experiments [3] . All time signals are sampled at 44100Hz, the STFT is computed with windows of 80ms (3528 samples) with a hop-length of 20ms (882 samples). This results in $n = 4097$ frequency bins and $m = 1501$ time frames in the STFT for $30s$ of raw audio signal. No smoothing is applied to the STFT, and we set M as the amplitude spectrogram.

The τ values are chosen among $\tau = 5, 10, 20$. We used one template for each piano note such that $r = 88$. Finally to fix the peak-picking threshold for H^{test}, two oracle strategies are used: 1) use the same threshold for all songs, and report results for the best value on the grid $[0.01 : 0.01 : 0.4]$ 2) perform transcription with a song-dependent threshold, optimized on the same grid. The first case corresponds to a scenario where the threshold is pre-trained for a category of recording (music genre, recording conditions) while the second case corresponds to a hand-tuned threshold for a specific song to transcribe.

We compare our method with the Attack Decay (AD) model presented in Section 3 which, to the best of our knowledge, is the current state-of-the-art for

[3] Python code to compute CNMF and reproduce all the experiments is available at `https://github.com/cohenjer/TransSSCNMF`

unsupervised/semi-supervised piano transcription. The results reported in Table 1 are the exact results from [3] (AD [3]), and the results of AD when applying our post-processing (AD*). In both cases transcription is performed on H^{attack} as defined in [3]. Despite our efforts we were unable to exactly reproduce the original AD scores. In particular the original AD paper introduces smoothness in several aspects: the data spectrograms are locally averaged, and peak-fusion is performed in the post-processing. Consequently, the AD* results enable comparison between the proposed CNMF and AD in the same pre/postprocessing conditions, while AD [3] are the best results achieved by Cheng *et. al.*. For completeness, we also report the transcription score from the state-of-the-art piano transcription network introduced in [8] which was trained on MAESTRO [7].

To measure performance, we compute a notewise score using the mir_eval [22] toolbox with a tolerance of 50ms. The offset detection problem is not tackled. Results are shown using only F-measure (F) and Accuracy (A) metrics (reported in percent), but full results including Precision and Recalls for all pianos are available in the complementary materials online.

4.2 Transcription Performance on MAPS

The MAPS dataset contains classical piano music pieces recorded with different pianos and conditions: a Yamaha Disklavier™in two settings 'ENSTDkCl' (EN1) and 'ENSTDkAm' (EN2), and six synthetic pianos 'AkPnBcht', 'AkPnBsdf' (AkB1-2), 'AkPnCGdD' (AkC), 'AkPnStgb' (AkS), 'SptkBGAm' (Sp), 'StbgTGd2' (St). For each piano/setting listed above, we train a template W^{train} using a rank-one CNMF as presented in Equation 7. Since there are many available single notes recordings in MAPS, we chose based on performance to use the Isolated notes (ISOL / NO) recorded at Medium intensity (M).

In a first experiment, we study the sensitivity of the proposed method to the selection of the convolution window size τ and the choice of a threshold δ tuned on the whole corpus versus on each song individually. We also compare our results to the Attack Decay model and the ByteDance supervised neural network. Results are shown in Table 1. From this experiment, we see that generally $\tau = 10$ performs best, and that the song-wise threshold gives better results.

In a second experiment, the templates for all other pianos are used to transcribe AkPnCGdD and ENSTDkCl to estimate the generalization capacities of the trained CNMF templates, see results in Table 2. Only CNMF with song-tuned threshold is shown and we set $\tau = 10$, to show the best results only.

Additionally, Table 3 reports the average running times when training the templates and performing transcription on the AkPnCGdD recordings. This test was run on a personal computer with AMD Ryzen 5 2600™ processor and 16GB RAM.

4.3 Generalization on MAESTRO

A natural question regarding CNMF templates is how well they can be used outside their training context without any domain adaptation. While results shown in Table 2 already provide a partial answer, we also tried to apply CNMF to the MAESTRO dataset. However, since no individual notes recordings are publicly available for MAESTRO, we used the templates learnt from MAPS. We transcribed 20 songs from the MAESTRO test set randomly chosen.

The results are quite poor: even when choosing song-wise thresholds, for all templates, CNMF does not reach above 59% in F-measure (test results are available in the supplementary materials). For comparison, the state-of-the-art with supervised deep learning techniques reaches above 95% F-measure on MAESTRO. Its performance on MAPS with data augmentation are also state-of-the-art, around 89% F-measure on EN1, despite the training/testing mismatch.

5. DISCUSSION

In light of the experiments conducted in Section 4, let us discuss the strengths of the proposed CNMF. It exhibits a significant improvement with respect to the Attack Decay model, which as far as we know is state-of-the-art for semi-supervised piano transcription. This is even more true when using the same pre-processing and post-processing for AD and CNMF, the former being in particular prone to unstable activations which were not observed in the latter. We may therefore affirm that the improvement in performance is indeed due to the CNMF model design. In other words, CNMF with a semi-supervised setting is an efficient piano transcription method. From numerical results, it seems that a convolution window size $\tau = 10$ is a good compromise between quality of transcription and transcription computation time.

The CNMF method does not perform better than the supervised state-of-the-art method we denoted as ByteDance DNN, which is expected given that this neural-network competitor is trained on MAESTRO which contains more than two hundred hours of perfectly aligned MIDI and audio piano recording of professional level. We still reach similar performances on some pianos such as EN1, EN2 and AkS. Nevertheless, the ByteDance DNN is not trained on MAPS contrarily to the proposed semi-supervised CNMF.

Moreover, the proposed semi-supervised setting only requires a handful of training dataset which are relatively easy to acquire. Indeed, only individual notes recordings are necessary, without any audio and MIDI registration. Compared to the very large amount of data currently required by state-of-the-art deep learning approaches, this is a huge advantage of the proposed approach applicable to any acoustic instrument with well-defined onsets readily available. Sadly our study is limited to piano transcription. A perspective of this work is to apply it to transcribe polyphonic instruments for which recording registered MIDI and audio is challenging.

Finally, while the performance does depend on the choice

			EN1		EN2		AkB1		AkB2		AkC		AkS		Sp		St	
thresh		τ	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A
global	CNMF	5	78	65	70	55	88	80	75	62	83	72	80	69	81	70	75	61
		10	**85**	**75**	**77**	**64**	93	88	87	78	91	84	**88**	**79**	89	82	84	74
		20	83	72	76	63	**94**	**89**	**87**	**79**	**92**	**86**	87	79	**90**	**83**	**86**	**77**
	AD*		81	69	68	53	66	50	71	56	60	43	67	51	64	47	67	50
song	CNMF	5	82	70	74	59	90	82	80	69	87	78	84	74	86	77	81	69
		10	**88**	**79**	**80**	**68**	**95**	**91**	**90**	**83**	94	89	**90**	**82**	**93**	**87**	89	80
		20	85	75	78	66	**95**	**91**	**90**	**83**	**94**	**90**	89	81	92	87	**89**	**81**
	AD*		82	70	69	54	68	52	73	59	61	45	69	54	66	50	70	54
AD [3]			82	70	-	-	-	-	-	-	85	74	-	-	-	-	-	-
ByteDance DNN [8]			**89**	**81**	77	65	**98**	**97**	**95**	**90**	**98**	**96**	87	77	**97**	**95**	**95**	**90**

Table 1. CNMF, AD, and the ByteDance supervised network performance with respect to the choice of hyperparameter τ and the choice of the peak-picking threshold, without training/testing mismatch for CNMF and AD. Only the first 30s of each songs were used. AD* uses the same pre/post-processing as CNMF. Tolerance is 50ms.

		EN2	AkB1	AkB2	AkS	Sp	St
AkC	F	74	77	77	70	74	77
	A	59	64	63	56	59	63
EN1	F	76	67	68	69	67	69
	A	62	50	52	53	52	53

Table 2. Transcription scores for CNMF, with training/testing mismatch.

	Training			Transcription
τ	5	10	20	10
Av. time	56s	193s	634s	239s

Table 3. Average computation time for a learning pattern (Training) or transcribing 30s of a song (Transcription) for semi-supervised CNMF. Results are reported for the AkPnCGdD piano in MAPS.

of a good activation threshold, CNMF still performs well using a global threshold over all songs in MAPS for each piano. Therefore extensively tuning the threshold hyperparameter is not essential to the success of CNMF here.

Despite these encouraging results, CNMF has a few issues which open interesting perspectives. First, it clearly has a significant generalization problem, or in other words, the learning stage overfits the training data. From Table 2, it appears that a mismatch between training and testing inside MAPS, while detrimental to transcription performance, is not as severe as a learning on MAPS and testing on MAESTRO. A tentative explanation is that the MAESTRO recordings are live performances with quite loud reverberation, while the MAPS recordings are drier. Looking for an audio transformation of the templates that minimizes recording conditions mismatch would therefore probably prove beneficial to generalize pre-recorded CNMF templates. Retraining a template library given few annotated data in the testing set could also be a possible solution. Whether this domain adaptation can be done fully blindly is still unclear however.

Second, despite performance not relying too much on the threshold level, the threshold selection method on the other hand is extremely important. Using a fixed threshold instead of the adaptive peak-picking drastically decreased performances in our early tests. But this also means that the post-processing of activations can be further improved using more involved technique than thresholding each note individually, and this research direction should not be overlooked if transcription performances of CNMF are to be further improved.

Third, for simplicity only one template for each note was used for the transcription phase. However, most instruments sound quite differently depending on how they are played. The proposed semi-supervised framework currently does not account for this timbre variation with amplitude or technique, and adapting the current method to make use of several templates per notes is an interesting research direction.

Finally according to the results shown in Table 3, computation time is rather large even in the testing phase. With the current implementation, real-time processing is therefore prohibited. Using a CNMF solver dedicated to Kullback-Leibler divergence or working on a more efficient rank-one CNMF solver than [19] could nevertheless drastically reduce computation time.

6. CONCLUSION

The state-of-the-art for automatic piano transcription is undeniably nowadays detained by deep learning techniques. However these methods rely on very large audio and symbolic registered dataset which are potentially very hard to obtain. In this work, we propose a competitive semi-supervised matrix factorization model which only requires labeled recordings of each individual notes. We show that when there is no mismatch between the training data and the test data, our approach performs significantly better than semi-supervised state-of-the-art approaches, approaching supervised deep learning performance. Therefore, we believe that using CNMF instead of NMF is an important step towards learning more reasonable frequency templates in low-rank approximation techniques for piano transcription or other similar tasks. Further works should however be devoted to adapt pre-trained templates to reduce generalization error. Improving the onset detection

method, allowing timbre variation in templates and reducing computation time are other important research directions. Finally, the proposed semi-supervised approach should be tested with other instruments than the piano and in a multi-instrument setup.

Acknowledgments

Jeremy E. Cohen and Axel Marmoret thank ANR JCJC LoRAiA ANR-20-CE23-0010 for supporting this work.

7. REFERENCES

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2018.

[2] E. Vincent, N. Bertin, and R. Badeau, "Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 109–112.

[3] T. Cheng, M. Mauch, E. Benetos, and S. Dixon, "An attack/decay model for piano transcription," in *ISMIR 2016-17st International Society for Music Information Retrieval*, 2016.

[4] L. Gao, L. Su, Y.-H. Yang, and T. Lee, "Polyphonic piano note transcription with non-negative matrix factorization of differential spectrogram," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 291–295.

[5] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[6] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[7] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations*, 2019.

[8] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution piano transcription with pedals by regressing onsets and offsets times," *arXiv preprint arXiv:2010.01815*, 2020.

[9] Y. Yan, F. Cwitkowitz, and Z. Duan, "Skipping the frame-level: Event-based piano transcription with neural semi-crfs," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[10] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2003, pp. 177–180.

[11] N. Bertin, "Les factorisations en matrices non-négatives : approches contraintes et probabilistes, application à la transcription automatique de musique polyphonique," Ph.D. dissertation, 2009. [Online]. Available: http://www.theses.fr/2009ENST0051

[12] P. Smaragdis, "Convolutive speech bases and their application to supervised speech separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 1–12, 2006.

[13] T.-M. Wang, P.-Y. Tsai, and A. W. Su, "Score-informed pitch-wise alignment using score-driven non-negative matrix factorization," in *2012 International Conference on Audio, Language and Image Processing*. IEEE, 2012, pp. 206–211.

[14] E. Benetos, A. Klapuri, and S. Dixon, "Score-informed transcription for automatic piano tutoring," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*. IEEE, 2012, pp. 2153–2157.

[15] D. Jeong and J. Nam, "Note intensity estimation of piano recordings by score-informed nmf," in *Audio Engineering Society Conference: 2017 AES International Conference on Semantic Audio*. Audio Engineering Society, 2017.

[16] D. Donoho and V. Stodden, "When does non-negative matrix factorization give a correct decomposition into parts?" in *In Advances in Neural Information Processing 16*, 2003.

[17] X. Fu, K. Huang, and N. D. Sidiropoulos, "On identifiability of nonnegative matrix factorization," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 328–332, 2018.

[18] V. Leplat, A. M. Ang, and N. Gillis, "Minimum-volume rank-deficient nonnegative matrix factorizations," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3402–3406.

[19] D. Fagot, H. Wendt, C. Févotte, and P. Smaragdis, "Majorization-minimization algorithms for convolutive NMF with the beta-divergence," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 8202–8206.

[20] A. Degleris and N. Gillis, "A provably correct and robust algorithm for convolutive nonnegative matrix factorization," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2499–2512, 2020.

[21] V. Emiya, N. Bertin, B. David, and R. Badeau, "MAPS-a piano database for multipitch estimation and automatic transcription of music," 2010.

[22] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014.

Defending a Music Recommender Against Hubness-Based Adversarial Attacks

Katharina Hoedt[1] **Arthur Flexer**[1] **Gerhard Widmer**[1,2]
[1] Johannes Kepler University Linz, Austria
[2] LIT AI Lab, Linz Institute of Technology, Austria
katharina.hoedt@jku.at arthur.flexer@jku.at

ABSTRACT

Adversarial attacks can drastically degrade performance of recommenders and other machine learning systems, resulting in an increased demand for defence mechanisms. We present a new line of defence against attacks which exploit a vulnerability of recommenders that operate in high dimensional data spaces (the so-called *hubness problem*). We use a global data scaling method, namely Mutual Proximity (MP), to defend a real-world music recommender which previously was susceptible to attacks that inflated the number of times a particular song was recommended. We find that using MP as a defence greatly increases robustness of the recommender against a range of attacks, with success rates of attacks around 44% (before defence) dropping to less than 6% (after defence). Additionally, adversarial examples still able to fool the defended system do so at the price of noticeably lower audio quality as shown by a decreased average SNR.

1. INTRODUCTION

Adversarial examples were previously reported in various fields of application (cf. [1–3]) as small perturbations of input data that drastically change the performance of machine learning systems. Since then, numerous attempts to make systems more robust and *defend* them against these attacks have been made (cf. [4–6]).

In previous work [7], a successful attack on the music recommender FM4 soundpark [8] was presented, inflating the number of times (perturbed) songs were recommended by the system. The paper failed, however, to provide an outlook on how this attack could be weakened and did not present a method to defend against the attack. The proposed attack exploited *hubness*, a problem of learning in high dimensions that leads to some songs being recommended very often and other songs being never recommended [9]. As the attack amplifies the negative effect of hubness on the recommendation of songs, finding a defence against this kind of attacks would contribute positively towards the fairness of a recommender system.

To explain the existence of hub points, it was previously shown that for any high but still finite dimensionality, some points are expected to be closer to the center of all data (or local center in case of multimodal data) than other points and are at the same time closer, on average, to all other points [10]. Such hub points appear in nearest neighbour lists of many other points, resulting in asymmetric neighbour relations: a hub y is the nearest neighbour of x, but the nearest neighbour of the hub y is another point a ($a \neq x$). One approach to repair asymmetric neighbourhood relations and hence mitigate the hubness problem is Mutual Proximity (MP) [11], which globally scales distances (i.e., considers neighbourhood information of *all* objects), and transforms the distance between two objects into a measure that captures how similar the neighbourhoods of these two objects are.

In this work, we aim at finding a defence against adversarial attacks that exploit the hubness issue present in various recommender systems. Our contribution in this paper is threefold: (i) for the first time we utilise the hubness-reduction method MP as a defence against adversarial attacks; (ii) in order to create a more difficult defence scenario for MP, we additionally incorporate knowledge of the defence into a modified adversarial attack; and (iii) we also investigate MP as a post-hoc defence, i.e., using MP to post-process recommendations issued by an attacked but undefended system.

2. RELATED WORK

Hubness was first described in Music Information Retrieval (MIR) [12], but is now acknowledged as another aspect of the curse of dimensionality, and hence a general machine learning problem [10]. It was not only demonstrated to be a relevant problem in audio-based recommender systems [8, 9] (which are our focus here), but also in recommenders based on collaborative filtering [13, 14] as well as in general multimedia retrieval [15].

To reduce the effect of the hubness phenomenon, various hubness mitigation techniques were previously proposed. Feldbauer and Flexer differentiate between methods that use *dimensionality reduction*, centring-based methods that *reduce spatial centrality*, methods that aim at *repairing asymmetric relations* (e.g., MP), or using entirely *different distance measures* [16]. We look at MP in particular in this work, as it was previously applied to data of the FM4 soundpark [9], has been shown to be very effective at reducing hubness in diverse datasets [16] and is applicable even for large amounts of data with efficient approximations [17].

Defences against adversarial attacks can broadly be di-

Copyright: © 2022 Katharina Hoedt et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

vided into approaches in which adversaries are either *reduced* or *detected* (cf. [4]) in order to make a system more robust [18]. One of the most notable approaches is adversarial training [6], where adversarial examples are included in the training procedure of a system. As this is a complex task due to the necessity of computing adversarial examples during training time, multiple variants of this defence were proposed more recently (e.g., [19, 20]) with a focus on efficiency.

Adversarial defence methods are generally viewed to be at an arms race with adversarial attacks, as a lot of defences are only considered purposeful until a new attack is proposed which the method fails to defend against (cf. [18]). In this work, we therefore try to apply a method for defending against adversaries that does not require full knowledge of the specific attack; the one limitation we work with, however, is that the attack needs to exploit the hubness phenomenon. Note also that the real-world recommender that was attacked in previous work [7] would not permit adversarial training as a defence method due to the nature of the system (cf. section 4.1), as no training in the sense of learning model-parameters is performed.

In this work, we build upon two approaches that were previously published; first, we use the attack scenario proposed in [7] in order to provide the setting for our defence. Secondly, we apply the hubness-reduction method introduced in [9] and investigate its suitability as a defence method against adversarial examples. This new application is investigated in this work for the first time; in addition to that, we introduce an adapted attack which incorporates knowledge of MP and is hence equipped with more knowledge about the defence to test the boundaries of the proposed defence method.

3. DATA

The data we use in this work consists of 5,000 songs from the FM4 Soundpark [1] , a music discovery site that provides a platform where (lesser known) artists can upload and present their music free of charge. Website visitors can listen to and download all the music at no cost. Songs are generally assigned to one or two out of six different genres [8]. The songs have a duration of at least 2 and a maximum of 22.5 minutes, with an average of 4.1 minutes.

4. METHODS

4.1 The Music Recommender

The system we adapt in this work is the audio-based recommender system integrated in the FM4 Soundpark [2] [8], which was previously attacked in related work [7]. To prepare the data for the recommender system, every audio file is converted to mono and re-sampled to 22.05 kHz. The central two minutes of a song are further transformed to Mel Frequency Cepstrum Coefficients (MFCCs) by applying a Short-Time Fourier-Transform (Hann window with a window size of 1,024 and a hop size of 512), a transformation to Mel-scale and finally using a discrete cosine transform to compress the results to 20 MFCCs.

After the pre-processing step every song is represented by a multivariate Gaussian, which is estimated on the respective MFCCs [8]. To approximate the similarity of two songs in the next step, the Gaussians describing them are used to compute a symmetrised Kullback-Leibler (KL) divergence (cf. [8]), i.e.,

$$D_{\mathrm{SKL}}(\mathcal{G}_x, \mathcal{G}_t) = \frac{\mathrm{KL}(\mathcal{G}_x||\mathcal{G}_t) + \mathrm{KL}(\mathcal{G}_t||\mathcal{G}_x)}{2}. \quad (1)$$

Here $\mathcal{G}_x$ denotes the Gaussian computed to represent a song x and KL is the KL divergence of two Gaussian distributions. The symmetrised KL divergence allows us to determine the k nearest neighbours of any song using D_{SKL} as a distance measure. A recommendation for a particular song consists of these k nearest neighbours, where k is set to 5 as in the real-world recommender [8].

Note that for this recommender, adding a new song x' requires the computation of its Gaussian $\mathcal{G}_{x'}$, as well as obtaining the symmetric KL divergence between $\mathcal{G}_{x'}$ and all other Gaussians in the catalogue. This also means that for our experiments we can pre-compute these elements without loss of generality (cf. [11]).

4.2 Hubness

The music recommender we look at in this work suffers from the consequences of hubness, leading to over a third of the songs in the catalogue never being recommended [9]. So-called *hubs* are often among the nearest neighbours of songs and are hence frequently recommended, leaving no room for *anti-hubs* in nearest neighbour lists, and thus anti-hubs are never recommended. Hubs are usually determined by their *k-occurrence* O^k, which is a measure describing the number of times a particular object (here: song) is within the k nearest neighbours of all remaining objects that are part of a dataset. In what follows, we use a *hub-size* of $5k$, i.e., for a song to be considered a hub it has to have a k-occurrence $O^k \geq 5k$. Note that the mean O^k across all objects is equal k, with O^k significantly bigger than k indicating a hub. Anti-hub songs never occur within the nearest neighbours of other songs, meaning they have a k-occurrence $O^k = 0$; all remaining songs, with $0 < O^k < 5k$, are *normal* songs [9]. Note once again that we let $k = 5$ in our experiments, resulting in a hub-size that equals 25.

4.3 Mutual Proximity

Mutual Proximity was previously proposed as a global scaling method improving the negative effect of hubness by repairing asymmetries of the neighbourhood relation between any two objects in a dataset [11]. The main idea is to transform distances to a likelihood that one object x is the nearest neighbour to another object t (given the distribution of all distances to object t), and to combine it with the likelihood of also t being the nearest neighbour of x. Only when both these likelihoods are high will MP also

[1] https://fm4.orf.at/soundpark

[2] Currently not accessible due to Adobe Flash Player's phaseout beginning of 2021.

achieve a high value. To estimate MP, we follow the approach proposed by [11] and use the empirical distribution to compute

$$\mathrm{MP}(d_{x,t}) = \frac{|\,\{j : d_{x,j} > d_{x,t}\} \cap \{j : d_{t,j} > d_{t,x}\}\,|}{n}. \quad (2)$$

Here $d_{x,t}$ is short for $D_{\mathrm{SKL}}(\mathcal{G}_x, \mathcal{G}_t)$ and corresponds to the symmetric KL divergence between the Gaussians of x and t; n is the total number of objects. We turn MP into a distance by defining $D_{\mathrm{MP}}(d_{x,t}) = 1 - \mathrm{MP}(d_{x,t})$.

4.4 The Attack

In [7], an attack originally proposed by Carlini and Wagner [2] for speech processing was used to compute adversarial examples for the real-world recommender system described above. The Carlini & Wagner (C&W) attack is a targeted white-box adversarial attack, meaning it requires full knowledge of a system under attack and aims at changing the output of the system to a predefined target t. In our case, the objects to be adversarially modified are audio files representing Soundpark songs, and the *adversarial perturbations* δ we wish to apply to these are modifications to the audio, in the form of additive noise, that are (hopefully) imperceptible. To achieve a desired distorted system prediction, which in our case is an increased number of times a particular song is recommended, we minimise a system loss with respect to the target output iteratively. In this modification of the attack, the target corresponds to a song within the dataset that is already a hub, and hence is often recommended. We simultaneously try to keep the added adversarial perturbation δ as small (or imperceptible) as possible by also minimising the squared L2-norm of a perturbation.

In order to compute an adversarial perturbation δ for a particular song, with the aim of increasing its recommendations, we therefore try to minimise a combination of a system loss and the norm of the perturbation iteratively. This results in an optimisation objective and an update formula for δ as follows, which we can realise with gradient descent:

$$L_{\mathrm{total}} = \|\delta_{ep}\|_2^2 + \alpha * D_{\mathrm{SKL}}(\mathcal{G}_{x+\delta_{ep}}, \mathcal{G}_t) \quad (3)$$

$$\delta_{ep+1} = \mathrm{clip}_\epsilon(\delta_{ep} - \eta * \mathrm{sign}(\nabla_{\delta_{ep}} L_{\mathrm{total}})). \quad (4)$$

Note that a perturbation δ is here subscripted with the current epoch ep, and its updates are performed based on the sign of the gradient $\nabla_{\delta_{ep}}$ (w.r.t. δ_{ep}) and the factor η. Updates are further clipped in each iteration to stay between $-\epsilon$ and ϵ. The system loss of the music recommender here is represented by the KL divergence (D_{SKL}), and $\mathcal{G}_x$ represents a Gaussian of a particular song x. The multiplicative factor α balances the focus of the optimisation between finding perturbations more quickly or less perceptible. For each x, the target song t is chosen to be its closest hub song. A perturbation δ is updated until the attack is successful (stopping criterion), i.e., song x has a $O^k \geq 25$, or until 500 update steps were made. The perturbation δ_0 is initialised with zeros.

After a successful attack, the number of times a song is recommended by the system is therefore higher than before. This could be exploited in systems like the recommender of the FM4 Soundpark, as users can directly contribute to the song catalogue and could try to submit a perturbed version of their song in order to manipulate the recommender.

4.4.1 Modified Mutual Proximity Attack

The attack described above uses knowledge about the song encoding and distance measure used by the recommender (via $\mathcal{G}$ and D_{SKL}), but not about the specific defence (MP) it will face. In order to create a more difficult defence scenario for MP, we additionally incorporate knowledge of the defence into a modified adversarial attack.

More precisely, we attempt to attack the system by (I) leaving the main objective L_{total} unchanged (cf. Equation (3)) and only adapting the stopping criterion of the attack. The stopping criterion decides if an attack is successful, which is now only the case if the k-occurrence is at least 25 *after* distances between all objects in a dataset are rescaled with MP. This is different to the original case, in which the attack was successful if the k-occurrence exceeded 25 *without* applying MP. In what follows, we call this attack adaptation $C\&W_{KL}^{mod}$. For the second adaptation (II), the stopping criterion is adapted in the same way; additionally, we change the objective L_{total} such that we minimise an approximation of the MP between a song x and its target t (cf. Equation (2)) instead of the KL divergence. This approximation is necessary as MP itself is not differentiable. Including this knowledge results in an updated L_{total} in Equation (3) of

$$L'_{\mathrm{total}} = \|\delta_{ep}\|_2^2 + \alpha * \widetilde{D}_{\mathrm{MP}}(d_{x+\delta_{ep},t}), \quad (5)$$

with

$$\widetilde{D}_{\mathrm{MP}}(d_{x,t}) = 1 - \frac{\sum_i \mathrm{bt}_{i,t}(x) * \mathrm{bt}_{i,x}(t)}{n}, \quad (6)$$

$$\mathrm{bt}_{i,t}(x) = \max(\tanh(d_{x,i} - d_{x,t}), 0). \quad (7)$$

Here $\widetilde{D}_{\mathrm{MP}}$ corresponds to an approximation of $D_{\mathrm{MP}}(d_{x,t})$, i.e., $\widetilde{D}_{\mathrm{MP}} \approx 1 - \mathrm{MP}(d_{x,t})$. The numerator in Equation (6) is the approximation of the numerator in Equation (2), and consists of summation over i, where i denotes all elements in our data catalogue. The constant n in the denominator denotes the total number of objects in the catalogue. The function bt is a differentiable approximation of the *bigger-than* function in Equation (2); here the function $\tanh$ denotes the hyperbolic tangent function, and $\max$ returns the maximum of its two inputs. Subsequently, this modified attack will be denoted by $C\&W_{MP}^{mod}$.

5. EXPERIMENTS

To allow reproducibility of the experiments summarised in this work, the code as well as all necessary attack-parameters are available on Github [3] .

[3] https://github.com/CPJKU/hub_defence

5.1 Parameters of Attacks

The adversarial attack proposed in [7] and our modifications require a set of parameters, which influence the number of successful adversarial examples as well as their quality. More precisely, the parameters we need to choose are the clipping factor ϵ, the multiplicative factor η which determines the step-size of the gradient updates, and finally factor α, which is responsible for controlling the focus of the attack on finding a higher number versus less perceptible adversaries. For the original attack, we tried to reproduce the results shown in [7] and used the proposed parameters, i.e. $\epsilon = 0.1, \eta = 0.001, \alpha = 25$.

For the remaining attacks, we performed a grid-search over various parameter combinations, and chose the settings in which we found the overall highest number of successful adversarial examples. For C&W$^{mod}_{KL}$, this corresponds to $\epsilon = 1.0, \eta = 0.001, \alpha = 25$; For C&W$^{mod}_{MP}$, we set $\epsilon = 1.0, \eta = 0.0005$, optionally with $\alpha = 100$. Note that this grid-search was done on the complete data base, since the white-box nature of the attacks requires full knowledge of the attacked system (here: all Gaussians). Additionally an attacker would naturally also have knowledge of the audio to be perturbed, hence a distinction into train and test data for parameter estimation, as is customary in most machine learning settings, is not necessary when evaluating such white-box attacks.

5.2 Mutual Proximity as a Defence

Before using MP as a defence, we investigate the vulnerability of the undefended recommender by applying the C&W attack used in related work [7]. The first line in Table 5.2 shows the result of this attack.

The columns in Table 5.2 depict the nature of the attack, the number of initial hubs (i.e., before the attack), the number of adversarial hubs (i.e., songs for which the attack was successful), and the number of songs for which the attack was not successful (# Non-hubs). The last two columns contain the average $\pm$ standard deviation of the Signal-to-Noise ratio (SNR) and the k-occurrence of successful adversarial examples. We manage to successfully attack around 44.1% of all files with an average SNR of roughly 39.0dB, which is similar to the results shown in [7]. Note that we use a subset of the data used in [7].

Next, we use MP to defend against future adversarial attacks by integrating it in the music recommender. We first rescale the distances between all original (clean) objects in our dataset with MP. Beyond what is shown in Table 5.2, let us briefly look at some additional information displaying the impact of applying MP on the data. Before rescaling our dataset has 1,663 (33.3%) anti-hub and 202 (4.0%) hub songs, with a maximal k-occurrence of $O^k = 393$. This means that one third of the song collection is never recommended without MP. After rescaling however, this is reduced to 408 (8.2%) anti-hubs and only 3 (0.1%) hubs, with a maximal k-occurrence of $O^k = 35$. To further advocate the usage of MP, we can also look at the average retrieval accuracy R^k as defined in [11], which increases from 43.7% before MP to 47.7% after MP.

After applying MP, we use the two adaptations of the attack in an attempt to find adversarial hubs for the defended system. Going back to Table 5.2, the second line shows that the success rate of the attack decreases from 44.1% to only 2.5% when we now try to minimise the KL divergence between a song and its target (C&W$^{mod}_{KL}$). Also the SNR of the successful adversaries decreases from on average 39.0dB to 28.3dB. If we use the second proposed adaptation (C&W$^{mod}_{MP}$), and minimise the MP instead, the SNR is higher (42.9dB), however we are only successful for a small percentage (0.4%) of all files (see third line). In an attempt to find a larger number of successful adversaries to test the boundaries of MP as a defence, we repeat the attack using the adaptation C&W$^{mod}_{MP}$ once more, yet leaving out the factor $\|\delta_{ep}\|_2^2$ restricting the norm of the perturbation (i.e., minimising $L'_{\text{total}} = \widetilde{D}_{\text{MP}}(d_{x+\delta_{ep},t})$). The result is shown in the last line of Table 5.2. Disregarding the perceptibility of a perturbation during the optimisation process leads to a slightly higher success-rate of 5.7%, but also in a low average SNR of 20.4dB. The perceptual differences of the different attacks can be examined in the supplementary material [4], where listening examples are provided. The song excerpts are chosen to represent good as well as bad examples for particular attacks in terms of their perceptibility. Note that perturbations with a SNR of above $40dB$ are hardly perceptible; SNRs between $40-20dB$ are perceptible, at least when compared to the original audio, but without noticeably changing the essence of the song. SNRs lower than $20dB$ tend to become clearly perceptible or even disruptive.

5.3 Post-Hoc Defence

Instead of integrating MP into the system directly, MP could also be applied post-hoc to rescale the distances between all files *after* an attack , and hence post-process its recommendations. While in a real-world scenario it should be preferable to directly integrate the defence into a system to make an attack as difficult as possible, we want to briefly show that MP could also reduce the impact of an adversarial attack on an undefended system. For each of the $2,206$ hub songs we found when attacking the undefended system, we apply MP after one such song is added to the (otherwise clean) dataset, and observe how its k-occurrence and therefore hubness is changed due to the transformation.

For $2,193$ out of the $2,206$ (i.e., 99.4%) files we manage to decrease the k-occurrence enough to revert them back to *normal* songs (i.e., $0 < O^k < 25$). In other words, MP could also be a successful post-hoc defence for the C&W attack on the music recommender.

6. DISCUSSION

In this work we examined a specific real-world system to investigate the suitability of MP as an adversarial defence, which is why we briefly want to reason about how our findings could generalise to other settings. A limiting requirement of this defence is, due to its nature, that the attack is

[4] https://cpjku.github.io/hub_defence

Adaptation	# Initial Hubs	# Adversarial Hubs	# Non-hubs	SNR	O^k
original	202 (4.0%)	2,206 (44.1%)	2,592 (51.8%)	39.0 ± 5.1	41.7 ± 23.0
C&W$_{KL}^{mod}$	3 (0.1%)	126 (2.5%)	4,871 (97.4%)	28.3 ± 6.6	25.9 ± 1.3
C&W$_{MP}^{mod}$	3 (0.1%)	18 (0.4%)	4,979 (99.6%)	42.9 ± 7.8	26.5 ± 1.5
C&W$_{MP}^{mod}$ (no norm)	3 (0.1%)	286 (5.7%)	4,711 (94.2%)	20.4 ± 9.2	26.1 ± 1.4

Table 1. Results of the (adapted) C&W attacks. The columns are the adaptation of the attack, the number of initial and adversarial hubs, the number of non-hubs after the attack, and the average SNR and k-occurrence of successful adversaries.

aimed at exploiting the *hubness* issue in some way. However, we expect the proposed defence not to depend on the exact computation of adversarial examples, and hence to be useful against approaches different from the C&W-like attack.

As previously mentioned, hubness has also been shown to be an issue in various different (recommender) systems, most importantly in systems based on collaborative filtering [13, 14]. In other words, also systems like these are susceptible to attacks exploiting hubness, and could potentially be suited for and benefit from the proposed adversarial defence. Note here once more that hubness in a recommender system can lead to unfair recommendations, as a significant part of data is never recommended, while a relatively small part of the data is recommended very often (cf. [9]); adversarial attacks that exploit hubness additionally amplify this effect on the fairness of recommendations, which is why a defence against them is crucial. However, as MP was shown to be able to reduce the hubness for various kinds of data (cf. [16]), we assume that the suitability of MP as a defence extends to a variety of different systems based on diverse data.

A further point for discussion is the potentially high computational complexity of the methods applied in this work. While the original definition of MP has quadratic complexity (w.r.t. the dataset size n), approximations that are linear in n were previously proposed to allow an application of this method even to large datasets (cf. [11, 17]). The high complexity also carries over to the modified adversarial attack, in which we therefore also used an approximation of MP. Nevertheless, the computational cost increases with growing dataset sizes, which is problematic in particular for the informed (modified) attack, as the MP needs to be computed in every iteration. The defence, however, requires additional computations of MP only if new data points are added to a catalogue.

Our work also connects to recent results establishing that, as the local intrinsic dimensionality of data increases, nearest neighbour classifiers become more vulnerable to adversaries [21]. Datasets are often embedded in spaces of higher dimensionality than is needed to capture all their information. The minimum number of features necessary to encode this information is called intrinsic dimensionality (see [22] for a recent review). It is well known that hubness also depends on a dataset's intrinsic dimension [10]. Future work should explore this possible new link of the concepts of adversaries, high dimensionality and hubness.

7. CONCLUSION

In this work, we have evaluated an existing hubness reduction method (Mutual Proximity) as a defence against an adversarial attack on a real-world music recommender. Before the defence, the attack is able to artificially inflate playcounts of songs at the expense of other songs never being played, resulting in a very unfair recommender. While the success rate of the attack is around 44.1% before the defence with MP, we decrease it to $0.4\% - 5.7\%$ after the defence, despite incorporating knowledge of the defence in the attacks. In addition to making it more difficult to find successful adversarial perturbations, the defence also forces the attack to result in more perceptible perturbations. As we defend a specific real-world recommender in this work, we also discussed how this could generalise to other systems, and reason that it could be a suitable defence against diverse kinds of hubness-related attacks in the future.

Acknowledgments

This research was supported by the Austrian Science Fund (FWF, P 31988). GW's work is supported by the European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme, grant agreement No 101019375 (*Whither Music?*). For the purpose of open access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

8. REFERENCES

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. of the 2nd International Conference on Learning Representations, ICLR*, 2014.

[2] N. Carlini and D. A. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *Proc. of the 2018 IEEE Security and Privacy Workshops, SP Workshops*. IEEE, 2018, pp. 1–7.

[3] B. L. Sturm, "A simple method to determine if a music information retrieval system is a "horse"," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1636–1644, 2014.

[4] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proc. of the 25th Annual Network and Distributed*

System Security Symposium, NDSS. The Internet Society, 2018.

[5] D. Jakubovitz and R. Giryes, "Improving DNN robustness to adversarial attacks using jacobian regularization," in *Proc. of the 15th European Conference on Computer Vision, ECCV*, ser. Lecture Notes in Computer Science, vol. 11216. Springer, 2018, pp. 525–541.

[6] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. of the 6th International Conference on Learning Representations, ICLR*, 2018.

[7] K. Prinz, A. Flexer, and G. Widmer, "On end-to-end white-box adversarial attacks in music information retrieval," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 93—-104, 2021.

[8] M. Gasser and A. Flexer, "Fm4 soundpark: Audio-based music recommendation in everyday use," in *Proc. of the 6th Sound and Music Computing Conference, SMC*, 2009, pp. 23–25.

[9] A. Flexer and J. Stevens, "Mutual proximity graphs for improved reachability in music recommendation," *Journal of New Music Research*, vol. 47, no. 1, pp. 17–28, 2018.

[10] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Hubs in space: Popular nearest neighbors in high-dimensional data," *Journal of Machine Learning Research*, vol. 11, no. 86, pp. 2487–2531, 2010.

[11] D. Schnitzer, A. Flexer, M. Schedl, and G. Widmer, "Local and global scaling reduce hubs in space," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2871–2902, 2012.

[12] F. Pachet and J.-J. Aucouturier, "Improving timbre similarity: How high is the sky," *Journal of Negative Results in Speech and Audio Sciences*, vol. 1, no. 1, pp. 1–13, 2004.

[13] P. Knees, D. Schnitzer, and A. Flexer, "Improving neighborhood-based collaborative filtering by reducing hubness," in *Proc. of the 4th International Conference on Multimedia Retrieval, ICMR*. ACM, 2014, p. 161.

[14] K. Hara, I. Suzuki, K. Kobayashi, and K. Fukumizu, "Reducing hubness: A cause of vulnerability in recommender systems," in *Proc. of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 815–818.

[15] D. Schnitzer, A. Flexer, and N. Tomasev, "A case for hubness removal in high-dimensional multimedia retrieval," in *Proc. of the 36th European Conference on Information Retrieval Research, ECIR*, ser. Lecture Notes in Computer Science, vol. 8416. Springer, 2014, pp. 687–692.

[16] R. Feldbauer and A. Flexer, "A comprehensive empirical comparison of hubness reduction in high-dimensional spaces," *Knowledge and Information Systems*, vol. 59, no. 1, pp. 137–166, 2019.

[17] R. Feldbauer, M. Leodolter, C. Plant, and A. Flexer, "Fast approximate hubness reduction for large high-dimensional data," in *Proc. of the 2018 IEEE International Conference on Big Knowledge, ICBK*. IEEE Computer Society, 2018, pp. 358–367.

[18] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020.

[19] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. P. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2019, pp. 3353–3364.

[20] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, "You only propagate once: Accelerating adversarial training via maximal principle," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2019, pp. 227–238.

[21] L. Amsaleg, J. Bailey, D. Barbe, S. Erfani, M. E. Houle, V. Nguyen, and M. Radovanović, "The vulnerability of learning to adversarial perturbation increases with intrinsic dimensionality," in *Proc. of the 2017 IEEE Workshop on Information Forensics and Security (WIFS)*. IEEE, 2017, pp. 1–6.

[22] F. Camastra and A. Staiano, "Intrinsic dimension estimation: Advances and open problems," *Information Sciences*, vol. 328, pp. 26–41, 2016.

Audio Metaphor 2.0: An Improved Classification and Segmentation Pipeline for Generative Sound Design Systems

Joshua Kranabetter[1], Craig Carpenter[1], Renaud Bougueng Tchemeube[2], Philippe Pasquier[2], and Miles Thorogood[1]

[1]University of British Columbia
[2]Simon Fraser University

ABSTRACT

Soundscape composition and design is the creative practice of processing and combining sound recordings to evoke auditory associations and memories within a listener. We present a new set of classification and segmentation algorithms as part of Audio Metaphor (AUME), a generative system for creating novel soundscape compositions. Audio Metaphor processes natural language queries from a user to retrieve semantically linked sound recordings from a database containing 395,541 audio files. Building off previous work, we implemented a new audio feature extractor and conducted experiments to test the accuracy of the updated system. We then classified audio files based on general soundscape composition categories, improved emotion prediction, and refined our segmentation algorithm. The model maintains a good accuracy in segment classification, and we significantly improved valence and arousal prediction models - as noted by the r-squared (72.2% and 92.0%) and mean squared error values (0.09 and 0.03) in valence and arousal respectively. An empirical analysis, among other improvements, finds that the new system provides better segmentation results.

1. INTRODUCTION

Soundscape composers aim at creating a type of electroacoustic music that is "characterized by the presence of recognizable environmental sounds and contexts, the purpose being to evoke listeners associations, memories, and imagination related to the soundscape" [1]. Figure 1 demonstrates the relationships of these sound design contexts on the continuum moving from realistic to abstracted. Computationally assistive tools for sound design and soundscape composition production focus on soundscapes trending toward the real end of this continuum.

Recent advancements of our Audio Metaphor (AUME) generative audio model expand the system's ability to produce sonic experiences with depth and clarity. As sound designers and soundscape composers use creative and technical strategies to communicate the sense of a place as perceived by a listener, an important factor of a soundscape or sound design includes valence and arousal in people's perception. Russel introduces these factors in a circumplex model that facilitates evaluation and analysis of a stimulus such as sound [2]. AUME expands on previous work in soundscape emotion recognition with a unique interface to modulate valence and arousal.

The system's most recent iteration draws on a database of 395,541 hand-tagged files. Updated segment description algorithms improve emotion recognition significantly while maintaining classification accuracy. Background-foreground segmentation and composition processes are automated through simple natural language queries in AUME. A smoothing algorithm now produces superior boundary delineation of the emotive audio signal while maintaining temporal resolution. This level of automation affords sound designers the ability to create long, complex soundscapes with a simple text query. These improvements, combined with the interactive speed of AUME, can provide state-of-the-art solutions in sound design and soundscape composition.

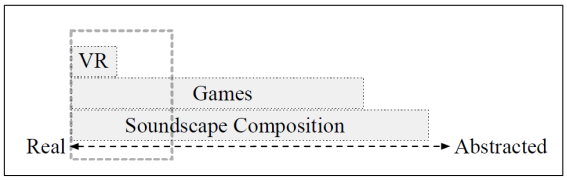


Figure 1. Continuum of ambience focus from real sounding environments, to more abstracted spaces. Production contexts range across the continuum. Figure from Thorogood et al. [3].

2. RELATED WORK

Generative soundscape models objective is to reduce the time-consuming practice in sound design of searching, importing, editing, sequencing, and arranging audio files from often unwieldy, massive online audio databases. Birchfield et al. describe a system that uses an adaptive user model for context-aware soundscape composition [4]. In their work, the system has a small set of hand-selected and hand-labeled audio recordings that were autonomously mixed with minimal processing. Similarly, Eigenfeldt and Pasquier employ a set of hand-selected and hand-labeled environmental sound recordings for the retrieval of sounds from a database by autonomous software agents [5]. In their work, agents analyze audio when selecting sounds to mix based on low-level audio features.

Copyright: © 2022 et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Like AUME, Teixeira et al. demonstrate a model that bypasses the tagging step through automating the browsing and retrieval of audio from large databases by employing Russel's widely used model of valence and arousal [6]. The MScaper system [6], integrated into Ableton Live DAW, similarly supports the adoption of the valence and arousal model for retrieving soundscapes. The valence and arousal model to facilitate soundscape generation also shares synergies with automated feature extraction developed by Music Information Retrieval systems (MIR). Bellisario and Pijanowski explore contributions of MIR to the emergent field of soundscape ecology and demonstrate how methodologies of automated feature extraction, sound classification and labeling using machine learning, and data visualization might be extended to soundscapes [7]. In further studies, Teixeira et al. have mapped how MScaper performs by using crowdsourced affective annotations from the Emo-Soundscapes dataset and mapped affective dimensions and low-level audio descriptors [6]. MScaper supports audio database navigation through a valence and arousal model, generating an adaptive soundscape according to emotional states. While AUME similarly employs a valence and arousal model, its unique composition engine utilizes a combination of Natural Language Processing (NLP), background and foreground classification, and segmentation to set it apart from related models in soundscape generation.

3. PREVIOUS WORK

Audio Metaphor (AUME) is a system for creating computer-assisted creation of soundscape compositions. Before this system update, the system performed better than random baseline in pleasantness, eventfulness, believability, and semantics [3]. The authors demonstrated AUME in the form of an art installation that derived natural language queries from Twitter. The queries were then used to generate soundscapes to reflect trending events in the social landscape. In previous research on Audio Metaphor, we established a framework for the system [8] (presented in Figure 2) and continue to describe each element throughout this section.

3.1 Corpus

The audio corpus consists of 395,541 sourced sound files from which we generate soundscapes. We save the audio files from source to disk to create the database. As described in the following sections, segmentation is required to obtain the short audio segments used for the automatic soundscape composition.

3.2 Crawler

The crawler is responsible for building the AUME database. It implements our feature extraction, classification, emotion recognition, and segmentation on a collection of audio files. The crawler finds an audio file, runs the pipeline, then saves the resulting segments into the AUME database. Each data entry in our database contains a file path to the raw MP3 audio data, associated file tags, and segment data. This segmentation and classification section, shown at the top of Figure 2, only runs once to build the AUME database.

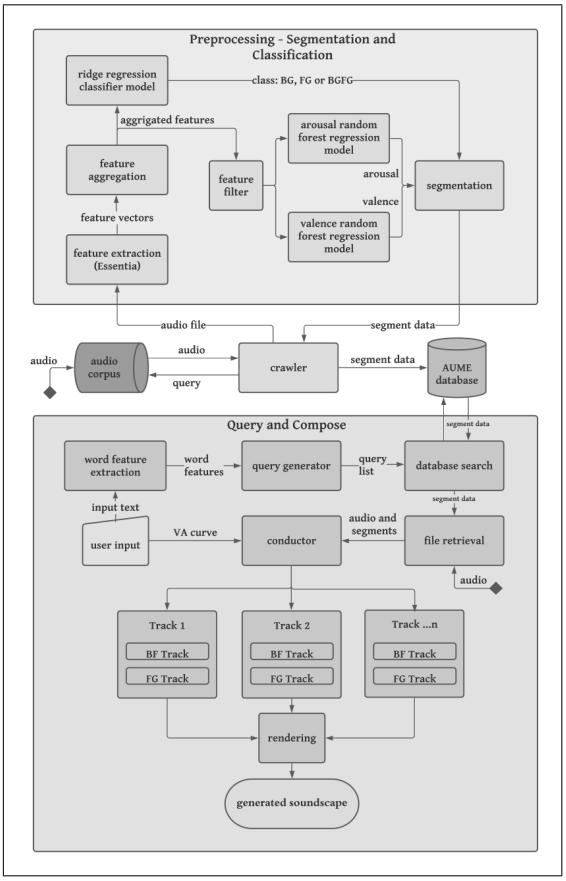


Figure 2. System diagram. Background, foreground, and background with foreground as BG, FG, and BGFG respectively.

After the crawler retrieves an audio file, the first job in the pipeline is feature extraction.

3.3 Feature Extraction

The audio feature set we generate in feature extraction contains spectral and perceptual audio descriptors of high and low levels. These descriptors attempt to model the human auditory system. This is desirable from a soundscape studies perspective, where the perception of the human listener is an important consideration. These features lay the foundation for the new classifier and emotion prediction models in Sections 7 and 8 respectively.

3.4 Feature Aggregation

We aggregate features using a bag of frames approach, where a signal is represented as a statistical distribution. The extractor provides feature data based on the input signal. We then divide that feature data into frames from which we can calculate the mean, standard deviation, skewness, variance, and their respective derivatives, and second derivatives. Frames do not sit end to end but overlap 50% with previous and following frames. To represent windows in an audio file, we group the output statistical data into segments. The segments can then be individually analyzed by our classification and emotion prediction algorithms.

3.5 Audio File Classification

An audio recording can be divided into the general classes of background, foreground, and background with foreground sounds. Sound designers and soundscape composers manually segment audio files into building blocks for use in a composition. We use machine learning to classify segments in an audio file automatically. As shown in Figure 2, the classifier uses extracted features to classify each segment in the audio file as foreground, background, or background with foreground. In previous work, our support vector machine (SVM) classifier achieved a true positive rate of 87.77%, a false positive rate of 12.22%, and a Kappa interrater reliability statistic of 0.8167. Audio file classifications are saved in the database to be used as the backbone in composition by the conductor. Next, we improve our composition further by adding emotion metrics for each segment.

3.6 Emotion Prediction

With the ability to interpolate Russel's circumplex model shown in figure 3, AUME retrieves audio segments evaluated on a scale of valence and arousal. Russel's model suggests all emotions are distributed in a circular space. High levels of valence correspond to pleasant sounds while low valence levels correspond to unpleasant sounds. Further, high levels of arousal correspond to exciting sounds while low levels correspond to calming sounds. Sound designers evoke emotion from a listener by dynamically controlling valence and arousal levels throughout a composition. We quantify levels of valence and arousal using machine learning for emotion prediction. The emotion prediction models use a subset of extracted features to predict valence and arousal for each segment in an audio file. We store the results in the AUME database for use in composition, a process we further explain in section 3.9. Previously, our emotion prediction models accounted for 62.9% and 85.5% for valence and arousal, respectively.

3.7 Segmentation

In segmentation, we aim to group background-foreground classified segments perceived as belonging to the same class. In the pipeline 2, segmentation is the final step before adding our segment data to the AUME database. We use segment information and background, foreground labels to filter, then conjoin like classes. We use a filtering algorithm to remove the noise of misclassified segments and negligible class appearances between two segments of the same class. The consequence of filtering is a resolution loss where detail can be reduced. In contrast, filtering can provide greater continuity from increased segment length, and therefore, more natural-sounding compositions. After we use the filtering algorithm, we finalize the segments by grouping adjacent segments belonging to the same class. The crawler finishes the audio prediction side of AUME by saving segment information into the database.

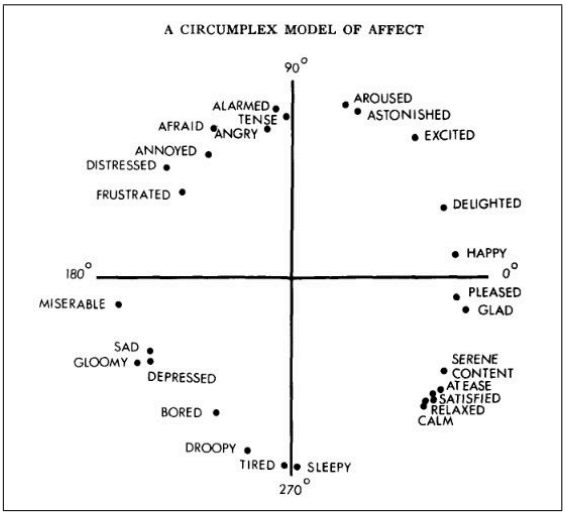


Figure 3. Common emotions displayed on the circumplex model of affect presented by Russel [2]. Arousal (eventfulness) occupies the vertical axis while valence (pleasantness) occupies the x axis.

3.8 Audio File Retrieval Using Natural Language Processing

The Audio Metaphor project uses an algorithm called SLiCE to process user input query text [8]. In the Natural Language Processing pipeline, common words listed in the Oxford English Dictionary Corpus are removed from the query, leaving only nouns, verbs, and adjectives. Words are kept in order and treated as a list. For example, with the word feature list from the natural language query: "The angry dog bit the crying man," "angry dog bit crying man," is more valid than "angry man bit crying dog." For a user input text with n words, the SLiCE algorithm constructs sublists of words of size 1 to n from the user input text to maximize the ability to match the appropriate sound segments. All unique sublists are put in a queue and used as search queries, starting with the longest first. When a search query returns a satisfactory result, all remaining queries that contain any of the successful word features are removed from the queue. The aim of the algorithm is to minimize the number of audio files returned and still represent all the word features in the list.

3.9 Conductor

The conductor is responsible for composing soundscapes using the audio supplied by the user query and associated class and emotion data. When a user queries the system, they can specify curves for valence and arousal to achieve a soundscape with their desired emotional characteristics. The conductor arranges segments that result from the search according to the specified curves. Next, the conductor mixes the tracks and renders the soundscape. Finally, AUME presents the resulting computationally generated soundscape to the user.

4. BUILDING A MODERN PIPELINE

To further research and develop Audio Metaphor, we move towards a state-of-the-art system to replace the YAAFE [9] audio feature extractor with Essentia [10]. Actively maintained by The Music Technology Group (MTG) of the University Pompeu Fabra in Barcelona, Essentia offers a more significant number of audio features. We expect Essentia to represent audio files with a higher degree of explanatory power and offer an opportunity to improve our predictive models - a process we explore in sections 7 and 8. Further, we investigate segmentation algorithms and their effects to tune the system for the best practical results.

5. CORPUS

The database for this project is a corpus of 395,541 audio files, a 96.8 fold increase from the original database of 4085 files. We extract files from the Freesound.org project using the Freesound API. The audio files from the Freesound project are uploaded in various formats (MP3, WAV, AIFF) and variable sample rates (130kbps to 196kbps) by individual users. They are accompanied by a collection of textual tags that describe the audio content's nature. The Freesound project processes the uploaded audio content to create normalized MP3 versions and metadata of sonic analysis. We extract audio files of duration ranging from two seconds up to 10 minutes. We do not use Freesound's sonic analysis as we tailor our own feature extraction to meet our specific needs.

6. FEATURE EXTRACTION

Essentia is an open-source C++/Python library for audio and music analysis. We use Essentia to sample corpus audio at full 22050Hz AIF format. In our bag of frames approach, we apply a frame size of 2048 samples and an analysis step of 1024 samples. We use the Blackman-Harris windowing function for spectral features. We use various statistics mentioned in section 3.4 to aggregate the features. To see all specific features we extract, see the Essentia documentation under the categories low-level stats and tonal stats [10]. This windowing configuration and subsequent analysis step result in high descriptive power for representing the texture and overall dynamics of the sound. Since we achieved good results with this method, we did not explore other window configurations.

7. SUPERVISED CLASSIFIER FOR SEGMENTATION

7.1 Corpus

We use the corpus created by Thorogood et al. [11], a curated collection of BF labeled sound files from the World Soundscape Project Tape Library database (WSPTL) [12]. The WSPTL contains five unique collections of soundscape recordings, with a total of 2545 individual sound files amounting to over 223 hours of high-quality, carefully selected recordings. The collections gathered between 1972 and 2010 are comprised of recordings from across Canada and Europe. Recording equipment included a Nagra IV-S field recorder and a pair of AKG condenser microphones. Collections are digitized and held online at Simon Fraser University [13].

The corpus is composed of 200 4-second samples from the WSPTL. Independent listeners confirmed 4-seconds was sufficient length for identifying the context of the sound. Further, the corpus is compact, so participants finished the study with minimum listening fatigue. Additionally, samples are short to preserve their class homogeneity for machine learning. The types of sounds cover the six soundscape categories defined by Schafer: natural sounds, human sounds, sounds and society, mechanical sounds, quiet and silence, and sounds as indicators. [14]. We mix the audio down to mono in favor of a higher degree of generality of the system. This compensates for recordings not obtained with similar high precision equipment or those recorded in mono.

Figure 4 shows the study's category agreement for the top 30 most agreed upon sounds by the study participants for each category, a data subset we call the BF90. A quantitative analysis of responses against the final corpus shows that participants agree on the category assigned to 92.5% (SD=3.6%) of the background samples, 80.8% (SD=9.5%) of foreground samples, and 75.3% (SD= 11.3%) of the background with foreground.

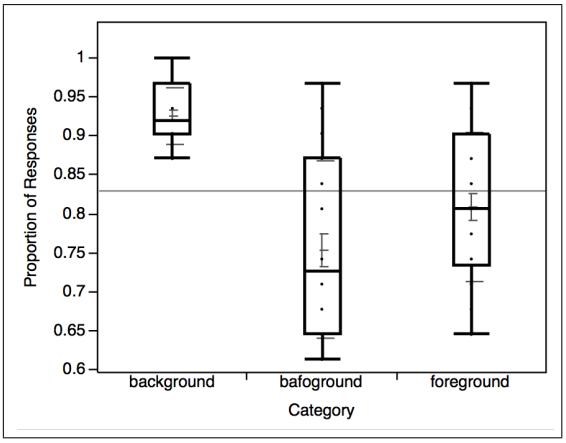


Figure 4. Box plots and mean lines for the agreement of labels for the corpus of background, foreground, and background with foreground recordings. The light grey line represents the overall mean agreement for the three classes. Figure from Thorogood et al. [11].

7.2 Model

Ridge regression is a type of regularized linear regression that incorporates techniques to reduce model complexity and prevent overfitting. In ridge regression, L2 regularization augments the cost function by adding a penalty equivalent to the square of the magnitude of the regression coefficients. This method performs well in cases where variables are highly correlated or the number of features exceeds the number of data points. We use the Scikit Learn [15] implementation of the ridge regression classifier which converts

binary targets to -1, +1, then performs multi-output regression. The predicted class is the output with the highest value. We use the default regularization strength of one.

7.3 Method

We perform an evaluation of the BF-Classifier using a repeated 10-fold cross-validation strategy on the BF90 dataset. This method randomly partitions the validation set into k = 10 equally sized sub-samples before iteratively testing the remaining sub-samples against each k-partition. We repeat the evaluation 10 times to reduce noise in the 10-fold evaluation.

7.4 Evaluation

Model prediction results are shown in Table 1 and a summary is shown in Table 2. The classifier achieves an overall true positive rate of 83.0%. An inter-rater reliability analysis using the kappa statistic determines the consistency of the classification. In this case, the kappa statistic of 0.743 shows good reliability of the classification results over the 10-fold validation.

	BG	FG	BGFG
BG	**23**	2	5
FG	0	**29**	1
BGFG	5	2	**23**

Table 1. Confusion matrix of SVM classifier for the categories background (BG), foreground (FG), and background with foreground (BGFG).

True Positive	83.0%
False Positive	9.30%
Kappa statistic	0.743

Table 2. Average true positive, false positive, and Kappa statistics of the ridge regression classifier.

8. REGRESSION FOR EMOTION PREDICTION

8.1 Corpus

We use the Emo-Soundscapes dataset for emotion recognition [16]. The dataset consists of 1213, 6-second long monophonic audio clips. Fan et al. curated 600 sounds from Freesound.org and mixed 613 audio clips from a combination of these. Additionally, the dataset contains a ranking for the perceived emotion in the 2D valence arousal space. They source rankings from 1182 trusted annotators from 74 different countries. The 1182 trusted annotators had a gold standard of 92.18% accuracy and provided a total of 69477 pairwise comparisons. We use the provided ratings, a translation of the rankings from 1 to 1213 mapped to a linear space between 1 to -1 inclusive.

8.2 Model

We use the Scikit Learn [15] implementation of random forest regression in prediction for both Valence and Arousal. It is a supervised learning technique that uses an ensemble method for classification and regression. Ensemble learning methods combine multiple algorithms to produce superior robustness than any of the component algorithms alone. In the case of the random forest, bootstrap aggregation (bagging) on each tree reduces the high variance of the decision tree. The result is a flexible model with reduced susceptibility to overfitting compared to a decision tree alone. It shows superior results compared to the support vector regression model provided by the emo-soundscapes data set and comparable results to deep learning methods [17].

8.3 Method

We randomly split the dataset into training and holdout sets; 80% for the training set and the remaining 20% for the holdout set. We perform feature selection to improve computational costs and improve model performance. We use 10-fold cross-validated recursive feature elimination on the training set. Most features were eliminated, with a 604 to 72 and 604 to 67 feature reduction in arousal and valence, respectively. Next, we tune hyper-parameters using five-fold cross-validation on the training set utilizing a range of values for maximum depth, minimum leaf samples, and minimum samples split. Finally, we perform the final evaluation on the holdout set.

8.4 Evaluation

We use Mean Squared Error (MSE) and R^2 to evaluate the performance of the models. Results expressed in Table 3 prove superior models when compared to previous research. The most recent previous model results using support vector machines (SVM) demonstrated 85.5% and 62.9% for arousal and valence, respectively [16], a significant improvement on previous research [18]. Our new model accounts for 92.0% and 72.2% of the variance for arousal and valence, respectively. Compared to deep learning methods, our random forest regression model achieves similar performance with a 2.8% improvement in arousal prediction and a 3.75% performance reduction in valence prediction. MSE values have also improved when compared to the most recent SVM models and are comparable to deep learning models. MSE of the arousal model has improved from 0.035 to 0.03 while the MSE of the valence model has waned from 0.078 to 0.09. We have achieved an increase in both valence and arousal model accuracy, but there is still a clear distinction in performance between the models. We estimate that while features like loudness can loosely predict arousal, valence has fewer strongly correlated descriptors, which results in a less accurate model.

Metrics	Arousal	Valence
R^2	0.920	.722
MSE	0.03	0.09

Table 3. Performance of Valence and Arousal models.

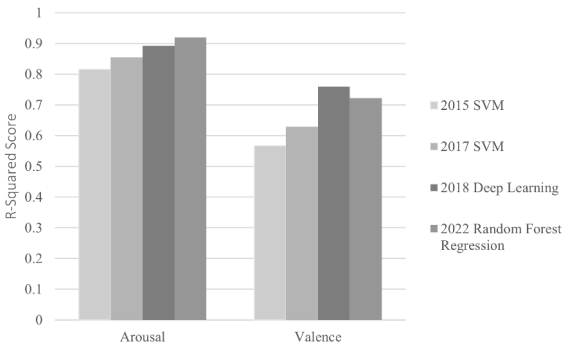


Figure 5. Previous and current predictive performance of valence and arousal models.

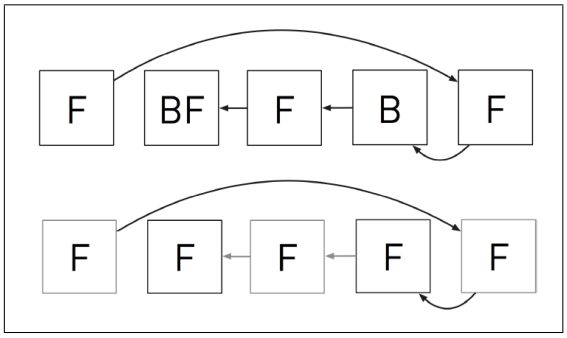


Figure 6. K-depth segmentation system jumping k = 3 then searching k - 1 and backtracking to relabel windows. Figure from Thorogood et al. [11].

9. SEGMENTATION

In this section, we take the results of previous segmentation work [11] and further refine the segmentation process to yield accurately labeled audio files. Thorogood et al. (2016) explore three segmentation algorithms: median filtering, k-depth lookahead, and Maximizing Posterior Probability (MPP). We empirically explore these algorithms in practice, then implement a solution to a prevalent misclassification problem.

9.1 K-Depth Look ahead

The K-depth algorithm traverses the list of segments and searches k-depth ahead of segment A to find the furthest instance B of the same class. If it finds a match, all instances of classes between the initial segment A and matched segment B are reclassified to match segments A and B. The algorithm then continues from the segment after B. If no match is found, we continue from the segment after A.

We evaluate the median filtering, k-depth lookahead, and MPP algorithms by segmenting 15 audio files and comparing the results. The median filtering algorithm smooths nicely but loses valuable foreground segments at all k values. In Figure 7, we show the results of median filtering for k values of 0 to 7. The MPP algorithm performed well overall, but ultimately the k-depth lookahead algorithm yielded the best smoothing while maintaining resolution. Similarly, Thorogood et al. [11] find the k-depth algorithm performs best, though we note that their evaluation window is 0.25 seconds while ours is set to 1.5 seconds. We set the k-value to 2 and give preference to foreground segments.

9.2 Margin Smoothing

In audio engineering, it is common to fade samples in and out. With the abundance of features we extract, the fade must have some non-trivial effect on the classification model. From the previous classifier, we observe an unusually high presence of foreground classification for the foremost and rearmost segments. Figure 7 shows a 44-second long example audio file that contains no foreground segments other than where the fade occurs: the margins. Every fade is likely misclassified as foreground because they emulate the high standard deviation in features commonly present in a foreground sound. We implement a margin smoothing technique for every file that uses a median filter to smooth the first and last segments classified as foreground. The median filter causes a loss of resolution at the margins but justifies itself by reducing overall classification error.

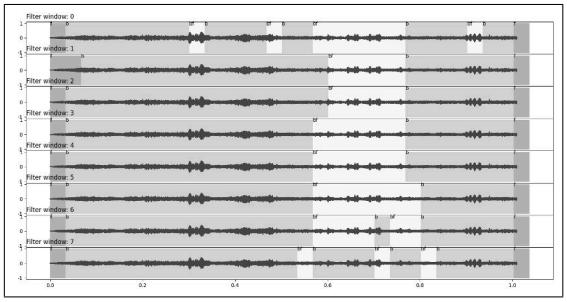


Figure 7. Median Filter Testing

10. RESULTS

To assess the performance of the new implementation of AUME, we compare segmentation between old and new systems. This comparison proves that the pipeline has been successfully reconstructed; we use it to evaluate the characteristics of each system. We extract background-foreground labeled segments from 15 audio files of various soundscape classifications using the results from the new system to compare them to the old system. The first comparison is illustrated in Figure 8, where the new system achieves identical results in the classification of this 44-second long audio clip containing sounds of the seashore. This comparison shows successful replication of the pipeline but fails to show any improvements or shortcomings on the new system. The second comparison is illustrated in Figure 9, where the new model shows slightly smoother results with only 10 segments compared to 12. This is representative of most other comparisons in that there are fewer total segments in the output of the new model. Additionally, Figure 9 demonstrates the effect of margin smoothing. The first segment produced by the old model is incorrectly classified

as foreground due to the fading effect mentioned in section 9.2. The new model corrects for this and correctly classifies the segment as background.

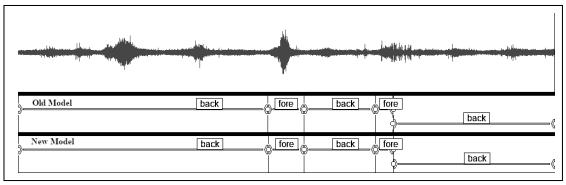


Figure 8. Labelled audio segments generated by previous (middle) and current (bottom) models for performance comparison for an audio file with identical results. The waveform (top) is displayed using the Audacity software.

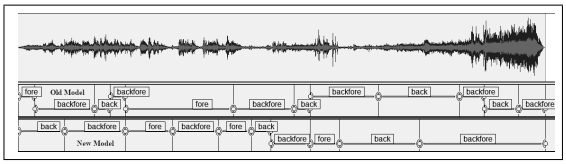


Figure 9. Labelled audio segments generated by previous (middle) and current (bottom) models for performance comparison for an audio file with divergent results. The waveform (top) is displayed using the Audacity software.

11. CONCLUSION

We describe a soundscape composition engine that chooses audio segments using natural language queries, segments and classifies the resulting files, processes them, and combines them into a soundscape composition at interactive speeds. This implementation takes user input to generate search queries and retrieves audio files that are semantically linked to audio files in the database. Sound designers can specify curves for valence and arousal to modulate the perceived emotion of the track over time.

The new AUME takes steps to improve generative soundscape composition by vastly expanding the database, implementing a robust classifier, drastically improving emotion prediction, and improving smoothness using segmentation. We expand the AUME database by 96.8 fold to increase system range and depth. System range is increased by covering a larger number of unique tags, while depth is increased by having more variations of tags already in the system. While the classifier model falls short by 3% when compared to the old classifier, it achieves similar results in practice 8. We improve prediction in arousal by 8.2% and valence by 15.6%. These improvements in emotion prediction offer the sound designer greater control in the depth and movement of emotion.

Audio Metaphor can be used to improve the productivity of sound designers by automating tedious audio database searches. Further, AUME automates the background, foreground segmentation process and composition. This level of automation affords sound designers and soundscape composers the ability to create long, complex soundscapes with a simple text query. When the length of the soundscape proves arduous for composers and designers, AUME offers a practical and efficient creative solution. Example soundscapes generated by AUME can be viewed online [1].

Acknowledgments

We would like to acknowledge the National Science and Engineering Research Council of Canada, and the Social Sciences and Humanities Research Council of Canada for their ongoing financial support. Additionally, we would like to thank Freesound.org for collecting and hosting creative commons licensed sounds.

12. REFERENCES

[1] B. Truax, "Soundscape, acoustic communication and environmental sound composition," *Contemporary Music Review*, vol. 15, no. 1-2, pp. 49–65, 1996.

[2] J. A. Russell, "A circumplex model of affect." *Journal of personality and social psychology*, vol. 39, no. 6, p. 1161, 1980.

[3] M. Thorogood, J. Fan, and P. Pasquier, "A framework for computer-assisted sound design systems supported by modelling affective and perceptual properties of soundscape," *Journal of New Music Research*, vol. 48, no. 3, pp. 264–280, 2019.

[4] D. Birchfield, N. Mattar, and H. Sundaram, "Design of a generative model for soundscape creation," in *in Proceedings of the International Computer Music Conference. International Computer Music Association*. Citeseer, 2005.

[5] A. Eigenfeldt and P. Pasquier, "Negotiated content: Generative soundscape composition by autonomous musical agents in coming together: Freesound." in *ICCC*, 2011, pp. 27–32.

[6] P. Teixeira, G. Bernardes, and M. Davies, "Fostering the database in audio production environments by affect soundscape retrieval."

[7] K. M. Bellisario and B. C. Pijanowski, "Contributions of mir to soundscape ecology. part i: potential methodological synergies," *Ecological Informatics*, vol. 51, pp. 96–102, 2019.

[8] M. Thorogood and P. Pasquier, "Computationally created soundscapes with audio metaphor." in *ICCC*, 2013, pp. 1–7.

[9] B. Mathieu, S. Essid, T. Fillon, J. Prado, and G. Richard, "Yaafe, an easy to use and efficient audio feature extraction software." in *ISMIR*. Citeseer, 2010, pp. 441–446.

[10] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, H. Boyer, O. Mayor, G. Roma Trepat, J. Salamon, J. R. Zapata González, X. Serra *et al.*, "Essentia: An audio analysis library for music information retrieval," in

[1] Online Sound Examples: https://digitalmedia.ok.ubc.ca/projects/aume-original/mixingExamples/

Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil.[place unknown]: ISMIR; 2013. p. 493-8. International Society for Music Information Retrieval (ISMIR), 2013.

[11] M. Thorogood, J. Fan, and P. Pasquier, "Soundscape audio signal classification and segmentation using listeners perception of background and foreground sound," *Journal of the Audio Engineering Society*, vol. 64, no. 7/8, pp. 484–492, 2016.

[12] B. Truax, "The world soundscape project," *WORLD SOUNDSCAPE PROJECT. Accessed May*, vol. 15, 2013.

[13] "The world soundscape project nbsp; to access the official wsp website, including its print publications, click here. to access the full wsp database with complete recordings, interviews, videos, etc. contact barry truax (truax@sfu.ca) for a guest password. the following introductory material is included here for ease of access." [Online]. Available: https://www.sfu.ca/~truax/wsp.html

[14] R. M. Schafer, *The soundscape: Our sonic environment and the tuning of the world*. Simon and Schuster, 1993.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] J. Fan, M. Thorogood, and P. Pasquier, "Emo-soundscapes: A dataset for soundscape emotion recognition," in *2017 Seventh international conference on affective computing and intelligent interaction (ACII)*. IEEE, 2017, pp. 196–201.

[17] J. Fan, F. Tung, W. Li, and P. Pasquier, "Soundscape emotion recognition via deep learning," *Proceedings of the Sound and Music Computing*, 2018.

[18] J. Fan, M. Thorogood, B. E. Riecke, and P. Pasquier, "Automatic recognition of eventfulness and pleasantness of soundscape," in *Proceedings of the Audio Mostly 2015 on Interaction With Sound*, 2015, pp. 1–6.

Scream Detection in Heavy Metal Music

Vedant Kalbag
Music Informatics Group
Georgia Institute of Technology, USA
vedant.kalbag@gatech.edu

Alexander Lerch
Music Informatics Group
Georgia Institute of Technology, USA
alexander.lerch@gatech.edu

ABSTRACT

Harsh vocal effects such as screams or growls are far more common in heavy metal vocals than the traditionally sung vocal. This paper explores the problem of detection and classification of extreme vocal techniques in heavy metal music, specifically the identification of different scream techniques. We investigate the suitability of various feature representations, including cepstral, spectral, and temporal features as input representations for classification. The main contributions of this work are (i) a manually annotated dataset comprised of over 280 minutes of heavy metal songs of various genres with a statistical analysis of occurrences of different extreme vocal techniques in heavy metal music, and (ii) a systematic study of different input feature representations for the classification of heavy metal vocals.

1. INTRODUCTION

Vocals in heavy metal music can be very different to those in other styles. Heavy metal vocalists use a variety of techniques, colloquially known as screams or growls, which are produced by modifying the length and shape of the vocal tract [1]. These screamed vocals serve one of two purposes: they are either low and beast-like to accentuate the aggressive, darker themes of heavy metal, or high and screechy, to stand out from the otherwise aggressive sounds of the distorted electric guitar [2]. In this paper we explore methods to detect and classify the type of vocal technique being used by a vocalist.

The automatic identification of different type of vocal techniques in heavy metal could, for instance, inform genre classification systems and aid music recommendation systems based on preference for a specific vocal type. Vocal detection for heavy metal music could also improve vocal extraction as well as (lyrics) transcription for this genre.

Nieto introduced the term 'Extreme Vocal Effects' or EVEs to describe the vocal styles present in heavy metal [3]. These EVEs fall into 3 main categories:

- *Growls*: Growls are common in death metal. They are very noisy and the fundamental frequency is rarely perceived. They are usually loud and produce a high amount of spectral variation [1, 4]
- *Fry Screams*: Fry screams are similar to growls, but are brighter and not as loud. They are produced by a series of irregularly spaced glottal pulses that are induced by inhaling or exhaling [5]
- *Rough Vocals*: Rough vocals are obtained by adding variations in the vocal tract to obtain a harmonically richer spectrum [1, 6]. This is much more common in rock than in metal (e.g., for bands such as *Foo Fighters* and *Breaking Benjamin*).

Figure 1 shows a sample spectrogram for each class. Distinct patterns in the low and mid fry scream can be observed that distinguish them from the other types of screams. The high screams occupy a higher portion of the spectrum as well. It is important to note that, in these examples, the mid fry scream appears to have lower frequency content than the low fry scream. This is because these are examples chosen from different vocalists, and the perceived type of scream varies according to factors discussed in Sect. 3.

Some subgenres of metal also involve sung or 'clean' vocals. In this paper, 'screams' and 'growls' will be used to describe the overall style of distorted heavy metal vocals, and 'clean' will be used to describe sung vocals. The term growl usually refers to the low pitched, rough sounds uttered by animals. Humans occasionally use growl-like voices to express strong emotions. Examples of 'growl' phonations have been seen across the genres of jazz, blues, gospel, samba, country and pop. In ethnic music, the growl is found in *umngqokolo* (the vocal tradition of the Xhosa people), and throat singing (Tuvan and Mongolian) [7]. However, in recent times growls are most strongly associated with metal vocals.

Extreme metal screams can be performed by either inhaling or exhaling which has a noticeable effect on the timbre of the sounds produced. However, in most modern metal, screams are produced by exhaling, and so our work will focus on these types of screams.

The remainder of this paper is structured as follows. After an overview of related work in Sect. 2, a new publicly available dataset is introduced in Sect. 3. We describe several benchmark systems for detection and classification in Sect. 4 and present the corresponding results in Sect. 6. The conclusion in Sect. 7 summarizes the main contributions in gives a brief outlook of future work.

2. RELATED WORK

While there exists, to the best knowledge of the authors, no previous work on the automatic categorization of heavy metal vocals, one related field is the detection of screams in

Copyright: © 2022 Vedant Kalbag et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

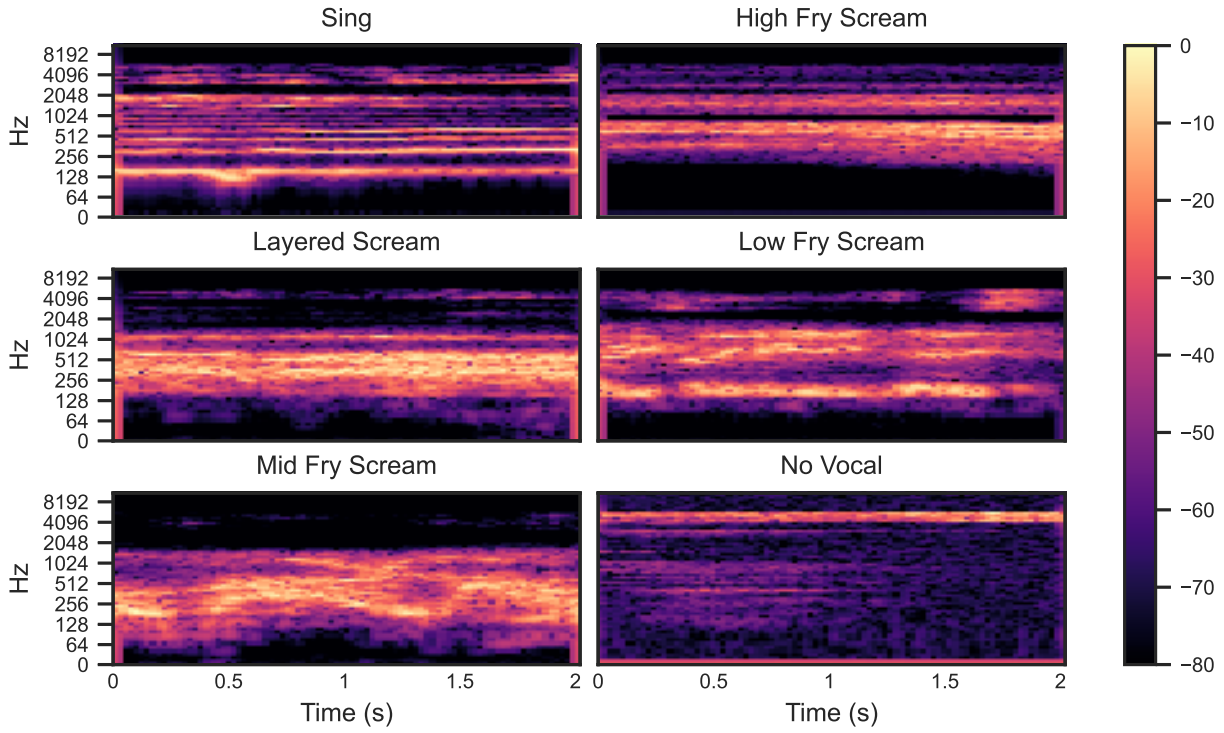


Figure 1. Example spectrogram representation of different screams.

urban environments.

In the previous section, we introduced different types of screams in metal music. Here, we will discuss past work with scream detection in general, followed by related work on screamed vocals in heavy metal music.

Prior work in detecting screams aims at the detection and localization of screams in urban sound, the detection of screams in subways, scream and shout recognition in noise, and scream detection for home applications. Various approaches were taken to achieve these tasks. Huang et al. used Mel Frequency Cepstral Coefficients (MFCCs) and Support Vector Machine (SVM) to classify blocks of audio captured from a microphone array into the two classes scream and non-scream [8]. Rabaoui et al. use a one class SVM to classify a sound into 9 categories, including scream, gunshot, explosion, door slam or dog barks, using features such as spectral centroid, spectral roll-off, zero-crossing rate, MFCCs and Linear Predictive Coding Coefficients (LPCCs) [9]. They also included the first and second derivatives of these features, but determined that they were not helpful in improving performance. Lafitte et al. used a deep neural network approach with MFCCs to detect shouted voice/screams in subway trains, and classify audio into shout, conversation and noise [10]. Other work in detecting screams in noise also uses MFCC and spectral entropy features with GMM classifiers to achieve this task [11–13]. The best performing of these methods was able to achieve equal error rates (EERs) of 0.3% and 0.8% under 0dB and -5dB signal to noise ratio (SNR) conditions. This approach, while useful in identifying screams in noisy conditions, cannot be translated well to detecting screams in music since the noise added was that of subway stations, trains and air conditioners.

Most work related to heavy metal vocals focuses on the physiology of screamed vocals [7, 14–16], their spectral properties [17], and exploratory acoustic feature analyses [7, 18]. There has been limited work on detecting and classifying the types of vocals present in heavy metal. Nieto uses k-means clustering to group different vocal styles into the three classes *Growl*, *Fry Scream*, and *Roughness* [1]. The dataset used consisted of labeled recordings of the 6 vocalists' screams. While this work was successful at grouping similar classes together, it could not predict the type of EVE present. Due to a lack of data with start and end times of vocal events annotated, a sliding window approach similar to Huang, where the scream detection algorithm is applied to every block in a sliding window to determine the start and end times of a scream [8] could not be implemented, and hence identifying when a scream occurs, or identifying what different kinds of screams are present within one file were not possible.

3. DATASET

Currently, there exists no publicly available dataset with annotated vocals for heavy metal. To enable this study, as well as to facilitate future research on this topic, we present the newly created *Metal Vocal Dataset* (MVD). This dataset consists of 57 songs from 34 bands and 47 albums. The list of songs can be found in the appendix. Most of these songs were released during the last two decades, since use of vocal effects beyond Mid Fry screams has increased in this period.

A playlist containing all the songs present in the dataset

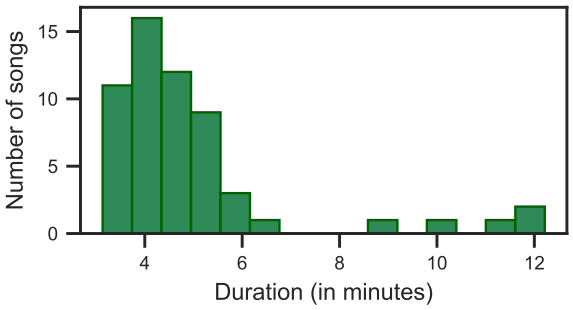


Figure 2. Distribution of dataset based on song length in minutes.

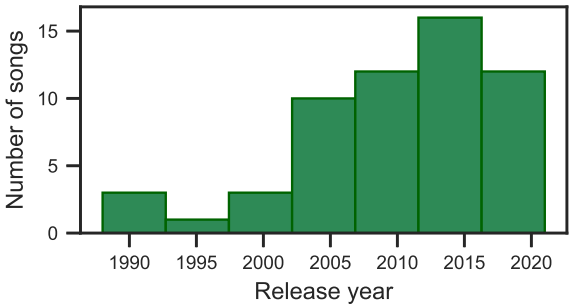


Figure 3. Distribution of dataset based on release year.

was created. [1] The distribution of the songs selected for the dataset based on the year of release is shown in Figure 3.

The annotations have been released under the MIT license and are available online. [2] The audio files themselves are not included, but can be retrieved using a script provided in the repository.

3.1 Data Selection

The songs selected were from genres such as death metal, groove metal, progressive metal, black metal, and metal core. The traditional subgenres of death metal, black metal and groove metal were included as they contain mostly one class of screams (mid fry screams), while modern subgenres such as metal core and progressive metal were chosen since a wide variety of vocal effects are used in these genres. The songs were selected with the aim to capture a wide variety in vocal styles and are listed in a playlist. [1]

3.2 Dataset Statistics

The distribution of the songs selected for the dataset based on the year of release is shown in Fig. 3. The increase for more recent years reflects the increased use of vocal effects beyond mid fry screams.

There are a total of 281.6 min of audio across the 6 classes (including the 'no vocal' class). The class distribution in the dataset is visualized in Fig. 4 based on the total time annotated in seconds. The Mid Fry scream is the largest part

[1] https://tinyurl.com/metal-vocal-dataset-playlist
[2] https://github.com/VedantKalbag/metal-vocal-dataset

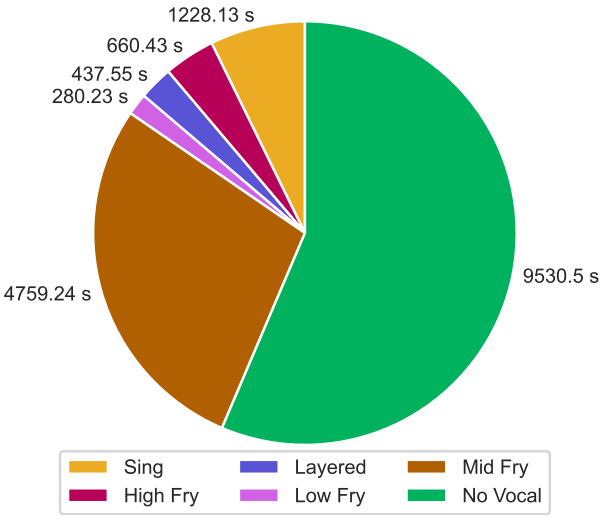


Figure 4. Distribution of dataset based on total time per class.

of the dataset; although the songs were selected carefully to contain all different classes, the Mid Fry scream is most prevalent in modern metal music.

3.3 Data Split

The data was split into 3 subsets for training, testing, and validation. This was done after division of audio files into 1 second blocks as described further in Sec. 4. Since the class distribution was heavily skewed towards blocks labeled 'no vocals', the dataset was undersampled to balance out classes. All classes that had more samples than the class with minimum samples were undersampled to the nearest thousand, for both the 3-class as well as the 6-class problem.

The data is accompanied by a recommended split into the subsets train, validation, and test (approx. 70:15:15). The data was split such that no band's songs are present in both the training and test/validation sets. Undersampling was applied before the split to balance the class distribution, as undersampling after the split would lead to considerably smaller test and validation sets. The blocks were first divided into an approx. 70:30 split, ensuring that no band was present in both subsets. This split at a band level was done to avoid overfitting any one vocalist/band and hence giving false results. The 30% split was then divided into two equal subsets at random. This was done because when restricting one band to be in either the test or validation set only drastically reduced the size of these sets, and would render them useless. In addition, a recommended split with imbalanced class distribution containing all data is provided as well.

3.4 Annotation Methodology

Since most screams in modern metal are variations of a fry scream, we have focused on these for our dataset. The variations are caused by a change in the shape and length of the vocal tract, where lengthening the vocal tract makes

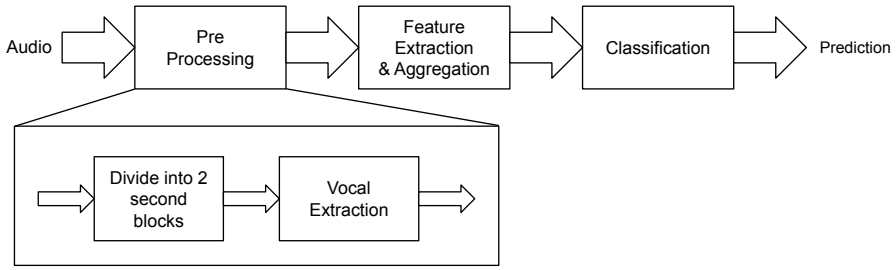


Figure 5. Block diagram of benchmark systems.

the scream sound lower, and vice versa. We have defined 3 fry scream categories based on the perceived sound: High, Mid, and Low. Thus, the vocal events were annotated with the following class labels: *Sing*, *High Fry* scream, *Mid Fry* scream, *Low Fry* scream, and *Layered* scream. The class labeled 'layered' contains combinations of 2 or more other classes simultaneously (e.g., Mid Fry screams and Sing, or both High and Low Fry screams).

These songs were manually annotated using *Sonic Visualiser* so that the maximum time difference between the start or end of a vocal event and the annotation is less than 0.5 s. The start and end points of the vocal event were localized visually based on the spectrogram of the audio file and validated aurally.

An important consideration is that the categorization of some screams is subjective, and two individuals may assign class labels differently. For example, a 'low-sounding' Mid Fry scream could be perceived as a 'high-sounding' Low Fry scream, and vice versa. As the main criteria for labeling the screams, the vowel characteristics of the sound were used. Typically, a Low Fry scream will have dark vowel characteristics (*/o/* or */u/*), a Mid Fry scream will have vowel characteristics similar to */a/*, and a High Fry scream will have characteristics around */e/* or */i/*. The labels were, thus, assigned based on how the scream sounded with respect to the perceived vowel characteristics; for instance, a scream with prominent low frequencies and vowel characteristics of */u/* or */o/* was labeled as a Low Fry scream.

4. BENCHMARK METHODS

A block diagram of the systems created as a benchmark for future work is shown in Fig. 5, and is described in detail in the following.

4.1 Pre-processing

The audio files were passed through the Spleeter source separation algorithm [19] to separate the vocals from the other components and then divided into overlapping blocks of length 2 s with a 1 s hop size. Each 2 s block is one observation to be classified. All audio files were resampled to a sample rate of 44100 Hz, normalized and downmixed to mono.

4.2 Input Representation

The baseline set of features consists of low level temporal and spectral features that are commonplace in Music Information Retrieval tasks. These features are: 13 MFCCs and Delta MFCCs, RMS, ZCR, Spectral Centroid, Contrast, Flatness and Roll-off (for a feature definition see [20]). These features were extracted using the Librosa python library [21], with a window size of 2048 samples and a hop size of 1024 samples. In addition, VGGish features [22] and the Log-Mel Spectrogram were extracted.

We divide these features into the following feature sets:

1. Feature Set 1: 13 MFCCs, Delta MFCCs, RMS, ZCR, Spectral Centroid, Contrast, Flatness and Roll-off
2. Feature Set 2: VGGish Features
3. Feature Set 3: 13 MFCCs and Delta MFCCs only
4. Feature Set 4: RMS, ZCR, Spectral Centroid, Contrast, Flatness and Roll-off
5. Feature Set 5: Log Mel Spectrogram

4.3 Feature Aggregation

All features in Feature Set 1 were aggregated by taking the mean and standard deviation across each audio block (with duration 2 s). The features in Feature Set 1, 2, 3, and 4 were all z-score normalized across the entire training set to return a feature vector with 0 mean and unit standard deviation. The mel spectrogram input was converted to log scale before use.

4.4 Classifiers

Two multi-class classifiers were used to classify each audio block based on the feature vector. The different classifiers used are a Support Vector Machine (SVM) and a Convolutional Neural Network (CNN). The CNN consists of 3 convolutional layers with dimensions 256, 512, and 1024, each followed by max pooling, respectively, 3 dense layers with dimensions 256, 64, and 16, and an output layer.

5. EXPERIMENTS

The system was tested for two different sets of labels: a 3 class problem (sing, scream, no vocal), as well as a 6 class problem (containing all the 5 labels from the dataset as well as no vocal).

Configuration	acc	bal-acc	f1
Feature Set 1 + SVM	82.20	82.10	82.18
Feature Set 2 + SVM	82.06	82.23	82.10
Feature Set 3 + SVM	77.12	76.95	77.21
Feature Set 4 + SVM	79.55	79.40	79.60
Feature Set 5 + CNN	**87.33**	**87.58**	**87.42**

Table 1. Results for the 3-class problem in Exp. 1 (values shown in %)

5.1 Experiment 1: 3-Class Problem

All scream classes are combined into a single class, resulting in the target set of classes *Sing*, *Scream*, and *No Vocal*. The following configuration were evaluated:

1. Feature Set 1 + SVM
2. Feature Set 2 + SVM
3. Feature Set 3 + SVM
4. Feature Set 4 + SVM
5. Feature Set 5 + CNN

5.2 Experiment 2: 6-Class Problem

As opposed to Experiment 1, Experiment 2 treats each scream class separately, resulting in the target set of classes *Sing*, *Low Fry*, *Mid Fry*, *High Fry*, *Layered*, and *No Vocal*. This experiment investigates the two best-performing SVM configurations and the CNN configuration from Exp. 1:

1. Feature Set 1 + SVM
2. Feature Set 2 + SVM
3. Feature Set 5 + CNN

5.3 Performance Metrics

The performance metrics used in this study are:

1. Accuracy: *acc*
2. Macro-Accuracy: *bal-acc*
3. Balanced F1 Score: *f1*

These metrics were computed with the sklearn python library [23].

6. BENCHMARK RESULTS

The results of both the 3-class and 6-class classification problem are presented below, followed by a discussion of the results. The results for a 3 class implementation, with blocks being classified into sing, scream and no vocal are compared to a 6 class implementation, where the audio block was classified into Sing, Low Fry scream, Mid Fry scream, High Fry scream, Layered screams and No Vocal.

6.1 Experiment 1: 3-Class Results

The results for each experiment are shown in Table 1 and the class-wise recall of each combination are shown in Fig. 6.

Figure 7 shows the t-SNE plot of Feature Set 1, and we can see a distinction between the 3 different classes, although some overlap between the classes *Sing* and *Scream*. We can make the following observations. First, combined Feature Set 1 outperforms Feature Sets 3 and 4 with a gap of roughly 5%. This is expected as these sets are subsets

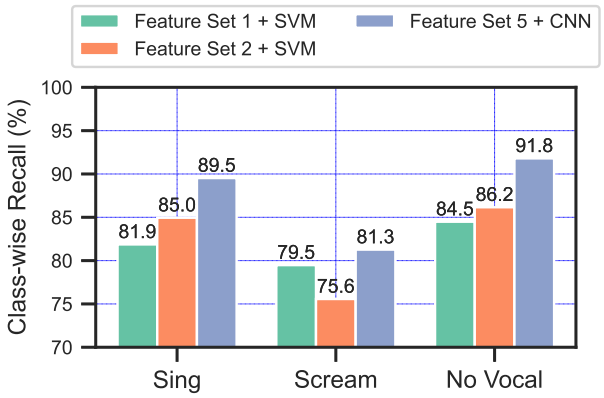


Figure 6. Class-wise recall for the 3-class problem.

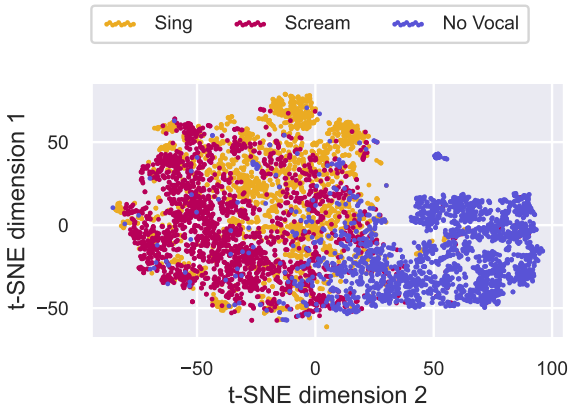


Figure 7. t-SNE projections of the feature space (Feature Set 1).

of Feature Set 1. Second, the combined Feature Set 1 and the VGGish Feature Set 2 show the best performance and perform similarly with recall above 82%. This means that the VGGish features, trained on a different task, contain a similar, semantically meaningful, information for classification as the combination of common baseline features. To a degree it is surprising that Feature Set 2 does not clearly outperform the traditional feature set as VGGish features have been shown to be powerful in music tasks such as musical instrument classification [24]. Third, the results show that the CNN with spectrogram input is able to detect the presence of screams with 87.6% balanced accuracy, which is notably higher accuracy than any SVM-based approach. It seems that the CNN is able to utilize the information in the spectrogram and is able to detect spectral patterns efficiently.

6.2 Experiment 2: 6-Class Results

The results of the 6-class problem are given in Table 2. We can observe that the performance is considerably lower for the 6-class problem with the two top-performing feature sets from Exp. 1. The VGGish features in Feature Set 2 seem to slightly outperform the low-level Feature Set 1. The CNN did not perform as well as the combination of

Configuration	acc	bal-acc	f1
Feature Set 1 + SVM	44.24	41.92	38.03
Feature Set 2 + SVM	**45.53**	**45.91**	**40.13**
Feature Set 5 + CNN	42.89	40.87	38.79

Table 2. Results for the 6-class problem in Exp. 2 (values shown in %)

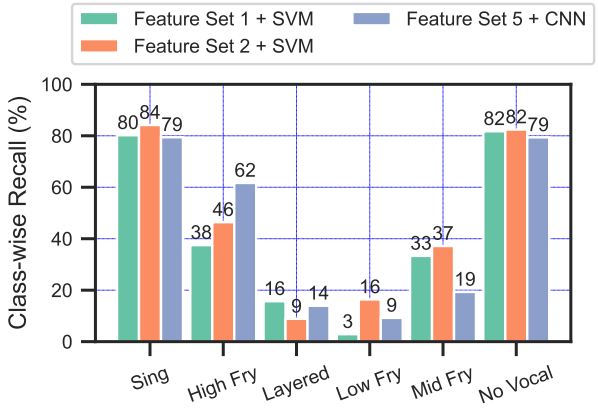


Figure 8. Class-wise recall for the 6-class problem.

VGGish features and SVM; the results of the CNN appear to be biased towards High Fry screams (see below).

Looking at the class-wise recall in Fig. 8, we observe that the systems could still identify the sung vocal and absence of vocals with high accuracy in the same range as the 3 class results shown above, however, they could not accurately distinguish between the different types of screams. We also see that the recall of the High Fry scream in the CNN is significantly higher than the other experiments, which is due to the classifier predicting most screams to be High Fry screams.

Investigating the confusion matrix in Fig. 9 gives us more details of the problem with the screams. We can see that several classes are being predicted incorrectly. Blocks labeled 'Layered' were often predicted as other classes, especially 'Sing' and 'High Fry' this could be because the layered class contains combinations of different classes, including the 'Sing' vocals. We also see that 'Low Fry' screams are often predicted as 'Mid Fry' due to the high degree of overlap between these classes in the feature space.

7. CONCLUSION

We introduced a new annotated dataset to aid and encourage further research in vocal detection in heavy metal music. Both the dataset and code have been made publicly available. While targeting scream detection, the dataset is also suitable for related tasks such as Vocal Activity Detection.

We presented a set of benchmark experiments on the automatic detection and classification of vocals in heavy metal music with the presented dataset. In these experiments, various temporal, spectral, and cepstral, and VGGIsh features were evaluated and compared with a CNN with log-mel spectrogram input.

In conclusion, with the dataset presented in this paper, we

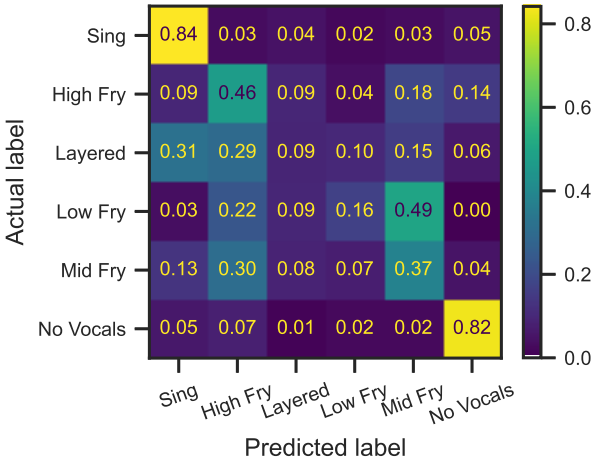


Figure 9. Confusion matrix for the 6-class problem (SVM).

were able to detect the presence of vocal events and classify them into sung vocal and screamed vocal with good accuracy. However, the same cannot be said for classifying the screams into the different types, as the different scream classes overlap within the feature space and cannot be separated easily. Thus, the dataset provides a new challenging task that can currently not be solved with satisfying results.

7.1 Future Work

There is anecdotal evidence online and within the heavy metal community for additional categories of vocal effects such as 'guttural vocals' and 'pig squeals'. Pending further investigation into this, we plan the extension of the dataset with additional audio files as well as extending the annotations to include these additional subsets of extreme vocal effects.

At present, the dataset has limited samples containing clean vocals sung over distorted instrumental sections, as most of the sections containing clean vocals in the songs used were also softer in nature. The dataset also has fewer samples of Low and High Fry screams (this is representative of their use in modern metal), and can be expanded upon by including further examples of these vocals.

8. REFERENCES

[1] O. Nieto, "Unsupervised clustering of extreme vocal effects," in *Proceedings of the 10th International Conference Advances in Quantitative Laryngology*, 2013, p. 115.

[2] N. J. Purcell, *Death Metal Music: The Passion and Politics of a Subculture*. McFarland, 2003.

[3] O. Nieto, "Voice transformations for extreme vocal effects," Master's thesis, Pompeu Fabra University, 2008.

[4] E. Smialek, P. Depalle, and D. Brackett, "Musical aspects of vowel formants in the extreme metal voice," *International Conference on Digital Audio Effects Conference*, pp. 1–8, 2012.

[5] C. T. Ishi, K.-I. Sakakibara, H. Ishiguro, and N. Hagita, "A method for automatic detection of vocal fry," *IEEE transactions on audio, speech, and language processing*, vol. 16, no. 1, pp. 47–56, 2007.

[6] E. Smialek, P. Depalle, and D. Brackett, "A spectrographic analysis of vocal techniques in extreme metal for musicological analysis," in *Proceedings of the International Computer Music Conference*, 2012.

[7] K.-I. Sakakibara, L. Fuks, H. Imagawa, N. Tayama, and others, "Growl voice in ethnic and pop styles," in *Proceedings of the International Symposium on Musical Acoustics*, 2004.

[8] W. Huang, T. K. Chiew, H. Li, T. S. Kok, and J. Biswas, "Scream detection for home applications," in *Proceedings of the Conference on Industrial Electronics and Applications*. IEEE, 2010, pp. 2115–2120.

[9] A. Rabaoui, M. Davy, S. Rossignol, Z. Lachiri, and N. Ellouze, "Improved one-class svm classifier for sounds classification," in *Conference on Advanced Video and Signal Based Surveillance*, 2007, pp. 117–122.

[10] P. Laffitte, D. Sodoyer, C. Tatkeu, and L. Girin, "Deep neural networks for automatic detection of screams and shouted speech in subway trains," in *International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 6460–6464.

[11] M. K. Nandwana, A. Ziaei, and J. H. Hansen, "Robust unsupervised detection of human screams in noisy acoustic environments," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2015, pp. 161–165.

[12] J. Pohjalainen, P. Alku, and T. Kinnunen, "Shout detection in noise," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2011, pp. 4968–4971.

[13] N. Hayasaka, A. Kawamura, and N. Sasaoka, "Noise-robust scream detection using band-limited spectral entropy," *AEU - International Journal of Electronics and Communications*, vol. 76, pp. 117–124, 2017.

[14] C. Eckers, D. Hütz, M. Kob, P. Murphy, D. Houben, and B. Lehnert, "Voice production in death metal singers," *Proceedings of the International Conference on Acoustics/35th German Annual Conference on Acoustics*, pp. 1747–1750, 2009.

[15] P. Ribaldini, "Heavy metal vocal technique terminology compendium: A poietic perspective," Master's thesis, University of Helsinki, 2019.

[16] A. Loscos and J. Bonada, "Emulating rough and growl voice in spectral domain," in *Proceedings of the International Conference on Digital Audio Effects*, 2004, pp. 49–52.

[17] M. Guzman, K. Acevedo, F. Leiva, V. Ortiz, N. Hormazabal, and C. Quezada, "Aerodynamic characteristics of growl voice and reinforced falsetto in metal singing," *Journal of Voice*, vol. 33, no. 5, pp. 803–e7, 2019.

[18] K. Kato and A. Ito, "Acoustic features and auditory impressions of death growl and screaming voice," in *9th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 2013, pp. 460–463.

[19] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, p. 2154, 2020.

[20] A. Lerch, *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. Hoboken: Wiley-IEEE Press, 2012.

[21] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[22] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, and others, "CNN architectures for large-scale audio classification," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2017, pp. 131–135.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] S. Gururani, M. Sharma, and A. Lerch, "An Attention Mechanism for Music Instrument Recognition," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Delft: International Society for Music Information Retrieval (ISMIR), 2019.

9. APPENDIX

The following songs were included in the dataset (Song No. Artist – Song Name):

1. Abbath – Ashes Of The Damned
2. After The Burial – Lost In The Static
3. Amon Amarth – Destroyer of the Universe
4. Amon Amarth – Live For The Kill
5. Amon Amarth – Twilight Of The Thunder God
6. Be'lakor – Venator
7. Behemoth – Ecclesia Diabolica Catholica
8. Behemoth – Bartzabel
9. Behemoth – Blow Your Trumpets Gabriel
10. Born of Osiris – White Nile
11. Cannibal Corpse – High Velocity Impact Spatter
12. Children of Bodom – Under Grass And Clover
13. Children of Bodom – Living Dead Beat
14. Children Of Bodom – Are You Dead Yet
15. Children of Bodom – Sixpounder
16. Children Of Bodom – Everytime I Die
17. Children Of Bodom – In Your Face
18. Dark Tranquillity – Lost to Apathy
19. Dark Tranquillity – Atoma
20. Death – Pull the Plug
21. Death – The Philosopher
22. Decapitated – Kill The Cult
23. Decapitated – Blood Mantra
24. Ensiferum – In My Sword I Trust
25. Enslaved – Caravans To The Outer Worlds
26. Godless – Deathcult
27. Gojira – Stranded
28. Gojira – Silvera
29. Immortal – Northern Chaos Gods
30. In Flames – Cloud Connected
31. Lamb of God – Memento Mori
32. Lamb of God – Laid to Rest
33. Lamb of God – Omerta
34. Lamb of God – Now You've Got Something to Die For
35. Lamb of God – The Faded Line
36. Ne Obliviscaris – Pyrrhic
37. Ne Obliviscaris – And Plague Flowers the Kaleidoscope
38. Nevermore – Born
39. Of Mice & Men – Bones Exposed
40. Of Mice & Men – Obsolete
41. Opeth – Blackwater Park
42. Parkway Drive – Carrion
43. Rings of Saturn – Senseless Massacre
44. Slayer – War Ensemble
45. Slayer – South Of Heaven
46. Slipknot – Psychosocial
47. Suffocation – Clarity Through Deprivation
48. Suicide Silence – No Pity for a Coward
49. Suicide Silence – Disengage
50. Suicide Silence – You Only Live Once
51. Suicide Silence – Slaves To Substance
52. Tesseract – Nocturne
53. Textures – Storm Warning
54. Textures – Old Days Born Anew
55. Thy Art Is Murder – Reign Of Darkness
56. Veil of Maya – Overthrow
57. Wintersun – Time

SMC-22 Paper Session 6

Ciaramella: A Synchronous Data Flow Programming Language For Audio DSP

Paolo Marrone
Orastron srl
paolo.marrone@orastron.com

Stefano D'Angelo
Orastron srl
stefano.dangelo@orastron.com

Federico Fontana
Università di Udine
federico.fontana@uniud.it

Gennaro Costagliola
Università di Salerno
gencos@unisa.it

Gabriele Puppis
Univesità di Udine
gabriele.puppis@uniud.it

ABSTRACT

Various programming languages have been developed specifically for audio DSP in the last decades, yet only a handful of industrial and commercial applications are known to actually use them. We assume that this is due to some common deficiencies of such languages, namely the tight coupling between syntax and computational model, which limits modularity, and the adoption of programming paradigms that are conceptually distant from conventional DSP formalism. We propose a new audio DSP programming language, called Ciaramella, based on the synchronous data flow (SDF) computational model and featuring a fully declarative syntax to address these issues. A source-to-source compiler which translates Ciaramella code to C++ and MATLAB programs has been developed. We have checked that our solution allows to naturally represent and correctly schedule highly-interdependent DSP systems such as Wave Digital Filters (WDFs) which would be hard to handle in current audio DSP languages.

1. INTRODUCTION

Audio programming languages raise interest in both the academia and the industry. They facilitate DSP systems programming by providing high-level abstractions to, e.g., efficiently manipulate data streams. There exist numerous audio programming languages and environments whose development dates back to the sixties, and they offer heterogeneous syntax, programming and computational models, and compilers or interpreters.

If the target is to develop computationally efficient audio code, then relevant and actively developed languages are FAUST [1], Max/Gen [2], Reaktor core [3], Soul [4] and Kronos [5]. FAUST is a mature language which adopts a purely functional paradigm, whose compiler performs extensive optimizations and translates to a wide set of plugin APIs. Gen from the environment Max is a visual programming language which combines graphical representation with textual instructions in a declarative fashion. Gen patches can be translated into VST plugins. Reaktor core, from the Reaktor music software studio, allows visual programming of DSP systems similarly to Max Gen but is not suitable for building plugins. Soul, a relatively recent project, provides a set of features for coding and running audio DSP systems; in particular, its language intentionally adopts a C++-like style improved with some instructions for working with data flows. Kronos, defined as a *metalanguage*, explores the declarative approach to facilitate the manipulation of signals, graphic visualization and just-in-time compiling. Conversely, SuperCollider and ChucK put emphasis on providing an immediate sound result, by featuring code interpretation and live coding.

Despite the existence of such domain specific languages, the music software industry tends to rely on general-purpose programming languages like C and C++ [6], although they force developers to pay attention to a number of details such as memory management, implementation techniques, compliance with plugin standards, and optimizations, which could rather be efficiently handled by domain-specific tools automatically. In our opinion, a potentially successful language should exclusively focus on the description of audio DSP systems using a limited set of domain-specific abstractions that are familiar to users and should be as modular and flexible as possible. At the same time, the compiler of such language should produce code that is easily embeddable in products that run on a wide range of platforms.

Unfortunately, none of the languages mentioned above meets all such requirements. For example, besides their differences, they share a syntactic rigidity by which it is impossible to describe delay-free feedbacks between components, whether built-in or composite (e.g., the subpatches of Gen): either a syntactical error is thrown or the feedback operation automatically implies the addition of a unit delay. In the former case, not only it is impossible to describe uncomputable systems containing delay-free loops, but also those computable ones containing loops in which delays are "hidden" inside a composite component. In the latter case, the implicit addition of a delay at a compositional level tightly couples the internal implementation details of each component to its external usage, which clearly violates encapsulation principles. Both arrangements prevent from easy de-

Copyright: © 2022 Paolo Marrone et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

scription and modular composition of multi-directional structures such those found in WDFs [7].

These practices are documented in the recursive composition section in the FAUST manual [1], in section 4.10 of the guide to Reaktor 6 core [2], in the feedback loops section of the SOUL documentation [3], in the History command section of Max/Gen [4] and in [5] for Kronos.

The proposed language addresses this problem at a semantic rather than syntactic level. It provides an unconstrained syntax based on a fully declarative approach and a dataflow block-based representation which should be familiar to DSP programmers. The compiler backend first *flattens* described systems into a representation consisting only of built-in components and interconnections, and then only at such a late stage evaluates and ensures their computability. Also, the underlying semantics of Ciaramella leverage the Synchronous Data Flow (SDF) framework which has been extensively researched [8] but not yet directly or fully adopted by current audio languages, despite allowing to naturally represent data flows and the mathematical operations on them. Finally, the compiler produces generic C++ and MATLAB code with no external dependencies. Ciaramella, however, is still at an experimental stage and it is far from being feature complete.

The paper is organized as follows. Section 2 recalls some theoretical facts about Synchronous Data Flow, reporting the current state of art of the SDF programming languages and showing how the SDF model fits the DSP domain. Section 3 describes the language concepts, abstractions, and syntax, furthermore showing some basic examples. Section 4 describes the compiler developed for Ciaramella, by focusing on the internal representation of the process topology, optimizations, computability verification, scheduling, and output code generation phases. Section 5 shows the implementation in Ciaramella of a simple WDF low pass filter.

2. SYNCHRONOUS DATA FLOW MODEL

SDF [8] is a restriction of Kahn Process Networks (KPN) [9]. It provides a set of primitives for describing a network of independent processes (also called *actors*) that communicate with each other via unidirectional FIFO queues emitting and consuming data values, called *tokens*. An *execution* is ruled by few constraints:

- writing (on a queue) is non-blocking;
- reading (from a queue) is blocking;
- reading implies token consumption;
- a queue can be written by only one process;
- a queue can be read by only one process;
- the number of tokens read and written by each process per queue per execution is known in advance.

Interestingly, the queues can be initialized with some tokens before the first execution. This corresponds to introduce a communication delay between two actors. An SDF *state* is defined as the number of tokens stored in every queue of the network.

The last rule differentiates SDF from KPN, and it is of crucial importance since it permits static scheduling at compile time [10] of the network for a correct sequential, or even parallel, execution. An SDF schedule is defined as a sequence of actor firings repeated periodically, and each repetition is called *period*. A periodic schedule is a finite schedule that invokes each actor at least once and does not change the SDF state: this guarantees that the tokens are not accumulated in the queues. Finally, a valid schedule is a periodic schedule which does not cause a graph deadlock, i.e. an actor planned for execution is not fireable if the input is unavailable. If, for a SDF graph, there is at least one valid schedule, the graph is defined to be *consistent*.

The SDF model is inherently multi-rate since the actors can produce and read different numbers of tokens at each firing. An actor generally performs simple operations like sum, multiplication, division, where it consumes two input tokens and produces one output token. However, *composite* actors have been defined [11] for modularity and compositional modelling, too. They encapsulate an SDF graph and act like normal actors, featuring readability and allowing to handle more complex systems easily. Conversely, non-composite actors are said to be *atomic* and a network composed only by atomic actors is called *flat*, so that the process of substituting the composite actors with their internal graph is said *flattening*.

2.1 SDF Languages

The SDF model inspired the development of some related programming languages in the last decades. They are thought mainly for synchronous reactive systems, which continuously interact with the execution environment at the speed imposed by it. The most relevant are Lustre [12], SIGNAL [13] and Esterel [14], all born in the eighties. Lustre gained some commercial success for some critical systems, too [15]. They all allow for the description of an SDF network in terms of actors and connections; most importantly, they provide temporal operators for accessing past values (delay) and for setting the initial value of the communication queues. Barkati *et al.* [16] made an especially important work for our study, since they analysed 10 programming languages, 5 of which are audio-specific and 5 SDF-specific, including the aforementioned ones. By comparing their syntax and expressiveness while implementing a digital oscillator, they showed, e.g., how

[1] https://faustcloud.grame.fr/doc/manual/index.html#recursive-composition

[2] https://www.native-instruments.com/fileadmin/ni_media/downloads/\manuals/REAKTOR_6_Building_in_Core_English_2015_11.pdf

[3] https://github.com/soul-lang/SOUL/blob/master/docs/SOUL_Language.md#feedback-loops

[4] https://docs.cycling74.com/max6/dynamic/c74_docs.html#gen_overview

correspondingly different notions of time were conceptualized: in SDF languages time is in fact more strictly an abstract logical notion, using multiple clocks to perform operations at different rates; on the other hand, audio languages adopt a less general and more practical approach directly using the sampling rate information. Concerning programming paradigms, while Esterel is imperative, Lustre and SIGNAL rather go for the declarative approach, in particular the equational and the relational ones, respectively. The SDF languages are descriptive à la VERILOG, leaving the scheduling of the processes to the compiler [10]. In summary, the study shows a close correlation between the audio and the synchronous domains by giving an overview of how they represent a typical DSP problem. We further investigate this relation by designing an audio programming language that fully adopts the SDF model.

For a detailed survey about synchronous programming of reactive systems check [17].

3. THE CIARAMELLA LANGUAGE

Ciaramella is an audio domain-specific programming language that adopts the SDF model for describing the DSP part of an audio plugin. Its design goals are:

- simple and unconstrained syntax;
- modularity;
- composability.

In order to meet them we chose the declarative paradigm, as opposed to the imperative one: this way, the order of the instructions is not relevant and a "variable" cannot be assigned more than once according to the static single assignment (SSA) form [18]. This makes the assignment operator semantically comparable to equivalence. Also, any expression in Ciaramella including identifiers always refers to streams/flows of data: for instance, the statement $a = b + c$ means $a_n = b_n + c_n$ at temporal sample n.

Our aim regarding the semantics of Ciaramella is to provide a minimal set of programming abstractions that is sufficient to represent SDF systems, as described in the following subsections.

3.1 Blocks, Ports, Connections

In Ciaramella there are three main components:

- Block: it represents an SDF actor and it encapsulates an operation. A list of input ports and a list of output ports are attached to it. Every block reads tokens from the input ports and writes tokens to the output ports.
- Port: it is a communication endpoint. It can be of input or output type.
- Connection: it defines a directed flow of data (tokens) between two ports. Together with the port, it is analogous to the SDF queue.

An output port can be shared by multiple connections. The current implementation of Ciaramella provides the following kinds of block.

3.1.1 Elementary Operation Blocks

Sum, subtraction, multiplication, division, sign change. They have 2 input ports and 1 output port except for the sign change/unitary minus which has 1 input port.

3.1.2 Variable, Numeric and Sample Rate Blocks

A variable block has 1 input and 1 output port and only works as a fork for data flows. A numeric block has only 1 output port and acts like a constant source of a given numeric value. The sample rate block is a special numeric block which outputs the sampling rate as dictated by the execution environment. It is accessible by the `fs` keyword at any point of the code.

3.1.3 Unit Delay Block

A unit delay has 1 input and 1 output port. Its operation is to store the nth input token and to write the $(n-1)$th one. In a pure SDF system a unit delay is obtained by pre-initializing a queue with one token. Operations on the connections in the SDF formalism are replaced by the unit delay block operation in Ciaramella in order to keep the language simple and the program structure clean. The two approaches are equivalent.

3.1.4 Composite Block

Composite blocks define sub-systems containing other blocks and connections and exposing a variable number of input and output ports. They are analogous to Nodes in Lustre [12] and subpatches in Max [2]. In a Ciaramella program there must always be at least one composite block which acts like a "main", and it is the only one that directly interacts with the execution environment.

3.2 Syntax

Ciaramella also aims at syntactic minimalism and essence by having a low number of keywords and adopting a conventional and simple design of the program structure. A Ciaramella program is made of a list of constant assignments with global scope and a list of composite block definitions with an internal local scope. The body of a composite block definition consists of a list of assignments. The following subsections provide more details.

3.2.1 Assignments, Expressions, Types

Assignments and expressions follow a C-like syntax. There are no explicit types, rather all values are meant to be in IEEE 754 floating point representation [19]. A composite block is defined by a list of output ports, its identifier, the list of input ports, and the body. The following example defines a stereo volume controller composite block:

```
y1, y2 = stereoVolCtrl(x1, v1, x2, v2){
    A = 0.8
    y1 = x1 * v1 * A
    y2 = x2 * v2 * A
}
```

`y1` and `y2` are the output ports; `stereoVolCtrl` is the name of the composite block; `x1`, `v1`, `x2` and `v2` are the input ports; the body is included between { and }. Within the body of the composite block all the declared output ports must be assigned and they can be used as variables within expressions. The input ports cannot be assigned since their value is set outside the block.

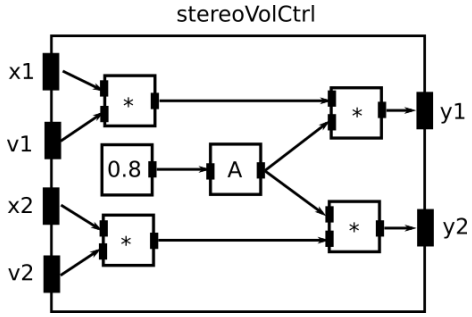


Figure 1: A representation of the `steroVolCtrl` composite block example.

Figure 1 contains a graphical representation of the `stereoVolCtrl` composite block. It shows how every expression creates a block, and sub-expressions are automatically connected to their parent expression; also, assignments determine connections between expressions and assigned variables.

3.2.2 Modularity and Composition

Defining composite blocks allows for reuse and modularization of the code. A composite block can be *instantiated* within another using a syntax resembling a function call in C. In the following example,

```
y = VolAttenuator(x1, x2) {
    t1, t2 = stereoVolCtrl(x1, 0.1, x2, 0.2)
    y = t1 + t2
}
```

the composite block `VolAttenuator` instantiates a `steroVolCtrl`. Intuitively, the first output port of `steroVolCtrl` gets connected to the input port of `t1` and the second one to `t2`, as shown in figure 2.

3.2.3 Unit Delays

Unit delays are the analogue of the *pre* operator in Lustre. It is fundamental to create explicitly computable loops within the SDF network. The expression `delay1(x)` returns x_{n-1}, i.e. the previous value. For example,

```
counter = delay1(counter) + 1
@counter = 0
```

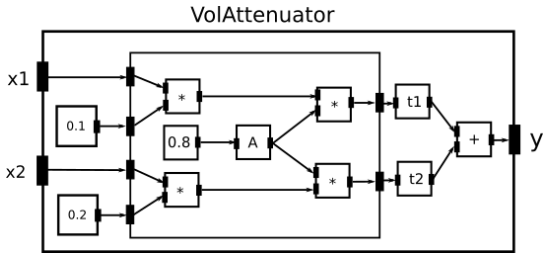


Figure 2: A representation of the `volAttenuator` composite block example.

implements a counter and it is equivalent to the recursive formula $counter_0 = 0$; $counter_n = counter_{n-1} + 1$ for $n \geq 1$. The second statement sets the initial value of `counter`.

By nesting unit delays, one can define a multiple delay as in

```
fib = fib_pre + delay1(fib_pre)
fib_pre = delay1(fib)
@fib_pre = 1
@fib = 0
```

which calculates the Fibonacci number and corresponds to

$$\begin{aligned} fib_{-1} &= 1 \\ fib_0 &= 0 \\ fib_n &= fib_{n-1} + fib_{n-2}. \end{aligned}$$

Note that these two examples are purely demonstrative are actually useless in audio applications.

3.3 Initialization Statement

We have shown how the `@A = expr` syntax sets the initial value of a variable. In particular, `expr` can be any valid Ciaramella expression, yet in this context the values of each variable in `expr` is evaluated as follows: if it is itself assigned an initial value expression, then this algorithm is applied recursively; otherwise, if there is a regular assignment expression in the current block, that is recursively used; in all other cases, the variable is necessarily external, hence its value is externally determined (it evaluates to 0 if the variable is a "main" block input). All delay operators are ignored.

In the following example

```
@A = 1 + B * C - x
@B = 3
B = x * 0.5
C = 2 + 5
```

`x` is an external audio rate signal, hence the initial expression of `A` evaluates to `@A = 1 + 3 * (2 + 5) - 0`.

3.3.1 Comments, End Statement

In Ciaramella a statement can be ended by either a semicolon or a new line. There are two kinds of comment: a classic single-line comment starting with a "#" and a

continuation comment starting with three dots, "`...`", lasting until the end of line, and which lets the statement continue on the next line.

4. ZAMPOGNA: THE CIARAMELLA COMPILER

We developed a small yet functional compiler for Ciaramella called Zampogna, sized ∼1000 lines of JavaScript code. The compilation process consists of several subsequent steps: the first ones are parsing and extended syntax validation, whose output is an Abstract Syntax Tree (AST). The AST is used to generate an intermediate representation graph (IG), corresponding to the SDF process network graph. Upon this graph, optimizations and static scheduling are performed. Finally, the compiler produces target code from the schedule.

4.1 AST to IG

The IG exactly corresponds to the set of components mentioned in Section 3.1. The compiler translates the AST to the IG by converting expressions to blocks and connections, recursively inserting sub-system instances while appropriately connecting them, and finally by flattening the resulting graph as described in Section 2. Figure 3 shows the flattened IG for the previous `VolAttenuator` example. Recursive block instantiation is not allowed.

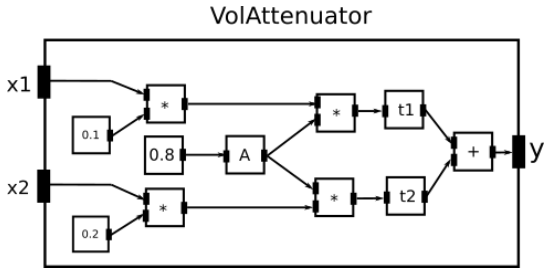


Figure 3: A representation of the `VolAttenuator` composite block example after the flattening operation.

The initialization statements (`@id = expr`) are not involved in the construction of the IG but, conveniently, the compiler uses them to produce another intermediate graph, called *Intermediate Initialization Graph* (IIG) which is executed only once, prior to the main IG, to produce initial values.

4.2 Optimizations

The compiler performs a series of optimizations over the IG. Some of them are common like dead code elimination, constant propagation, and copy propagation. Domain-specific optimizations are more interesting, particularly those based on value update rates. A typical audio application, besides processing audio signals, receives asynchronous input events at runtime. Usually the time distance between two of such events is significantly higher than the sampling period of the audio signals. In the SDF model, asynchronous events are treated as normal synchronous signals whose value is externally set. This keeps the domain coherent while providing to the programmer a unified view of signals, events, and constants. Indeed, every block of the graph emits one sample at each firing, regardless of its nature. For example, numeric constant blocks output the same token every time, audio rate blocks produce (potentially) different tokens at each firing and user control input blocks emit the same token until the user moves the associated knob or slider. Being aware of such characteristic of the blocks, we define the *update class* property which is necessary for the compiler to operate optimizations on the final code.

Four update classes are defined, ordered by increasing update frequency:

- **Constants class**. Numeric constants belong to this class.
- **Sample rate class**. The sampling rate value is set by the execution environment at runtime. This is typically done only once. `fs` block output falls into this update class.
- **Control rate class**. The output of the blocks representing user input controls are included in this class.
- **Audio rate class**. Input and output audio signals are part of this class, as well as the output of the `delay1` blocks.

The update class information is attached to the ports rather than the blocks since it is, conveniently, a property of the signals. It propagates to all nodes of the network starting from constants, `fs`, inputs and `delay1`. Recursively, the output port of a block inherits the highest rate of the input ports it depends on. Conversely, input ports inherit such property from the ports they are connected to.

Leveraging this information, the compiler makes sure that intermediate and output values are only updated when a change occurs or might occur. This behaviour is commonly known in other contexts as lazy evaluation of expressions.

4.3 IG Static Scheduling

After the optimizations over the IG are performed, the next phase is the scheduling of the atomic blocks for sequential execution. Such blocks need to be scheduled taking into account their mutual dependencies. In particular, it is possible to build a dependency graph (DG) starting from the IG: every directed arc of the IG, except for those starting from `delay1` blocks, define a dependency. If $A \rightarrow B$ indicates a connection from an output port of A to an input port of B, and if A is not a `delay1` block, then A must be executed prior to B; we say that B *has an instantaneous dependency on* A. Delay blocks are excluded because their output value is already known at the beginning of the n-th (current) iteration, as it was calculated at the $n-1$-th (previous) one. There can be multiple valid schedules and [10] investigates the

issue for multi-rate SDF graphs. Our case is simpler since we manage only a single synchronous rate and all blocks produce and consume only one token at each firing. Therefore, a valid schedule can be represented by a stack which can be filled by recursively removing the nodes that do not have instantaneous dependencies left and pushing them onto the stack.

If an instantaneous circular dependency path is encountered, the compiler throws an error. This arrangement is commonly known in the audio DSP literature as *delay-free loop* and is not computable. Unlike other languages, the ability to verify this condition at such a late stage, that is after having fully flattened the IG into atomic operations and having attempted scheduling on the full IG, frees us from all syntactical and semantic restrictions in representing instantaneous feedbacks. This allows for natural description of highly-interdependent DSP systems such as WDFs.

As an example, we implemented a wave digital filter in section 5: its unflattened graph appears to be uncomputable due to the presence of delay-free loops, while, when flattened, it is perfectly computable because delays are found within the composite blocks.

4.4 IIG Static Scheduling

The IIG is scheduled similarly to the main IG, but with an important difference: the `delay1` blocks are not excluded from the construction of the arcs of the DG as no previous value can be used. Therefore, in the IIG there cannot exist loops, no matter whether they contain delays or not.

4.5 Output Code Generation

Zampogna produces a C++ program with a VST2 wrapper, and a MATLAB script which is useful for fast prototyping. We used the doT JavaScript templating library [20] to accomplish this, which makes the code generation part of the compiler modular and easily extensible.

4.6 Compiler Options

In order to keep the syntax simple and the programs modular, some information can only be passed to the compiler via command line options. For example, the name of the "main" block and the list of input ports associated to user controls belong to this set of data.

```
Usage: zampogna.js [-i initial_block]
    [-c control_inputs] [-v initial_values]
    [-t target_lang] [-o output_folder]
    [-d debug_bool]
    input_file
```

4.7 Implementation

The implementation of the Zampogna compiler is available at https://github.com/paolomarrone/Zampogna. It has been developed for NodeJS and the only external dependencies are doT [20] for the output code generation and Jison [21] for the parser code. It comes with a few examples, like the WDF from Section 5.

5. A WAVE DIGITAL FILTER IMPLEMENTATION

As a case study we implemented a low pass transfer function using a WDF [22]:

```
pi = 3.141592653589793
b, R0 = wdf_resistor(a, R) {
    b = 0
    R0 = R
}
b, R0 = wdf_capacitor(a, C) {
    b = delay1(a)
    R0 = 0.5 / (C * fs)
}
b = wdf_voltage_source_root(a, E) {
    b = 2 * E - a
}
bu, bl, br, R0 = wdf_3port_series(au, al, ar, Rl, Rr) {
    bl = al - Rl / (Rl + Rr) * (al + ar + au)
    br = ar - Rr / (Rl + Rr) * (al + ar + au)
    bu = -(al + ar)
    R0 = Rl + Rr
}
y = lp_filter(x, cutoff) {
    fc = (0.1 + 0.3 * cutoff) * fs
    C = 1e-6
    R = 1 / (2 * pi * fc * C)

    bR, RR = wdf_resistor(aR, R)
    bC, RC = wdf_capacitor(aC, C)
    bV = wdf_voltage_source_root(aV, x)
    aV, aR, aC, Rp = wdf_3port_series(...
        bV, bR, bC, RR, RC)
    @aC = 0

    y = 0.5 * (aC + bC)
}
```

which can be compiled by the following command

```
zampogna.js -i lp_filter -c cutoff
    -t cpp lp_filter.crm
```

The composite block `lp_filter` instantiates other composite blocks, which mutually exchange signals by recursively creating loops. The compiler does not add implicit delays: the only unit delay is specified in `wdf_capacitor`. Even if at a first glance it may seem that the WDF gives rise to delay-free loop errors, indeed the system is found to be explicitly computable thanks to the flattening operation over the IG and the global scheduling. Figure 4 shows a simplified graphical view of the unflattened IG of `lp_filter`.

This case study shows how Ciaramella is able to natively represent a class of DSP problems, such as WDFs, that is known to be cumbersome to program in any other audio programming language. Furthermore, it demon-

strates its high modularity and composability characteristics.

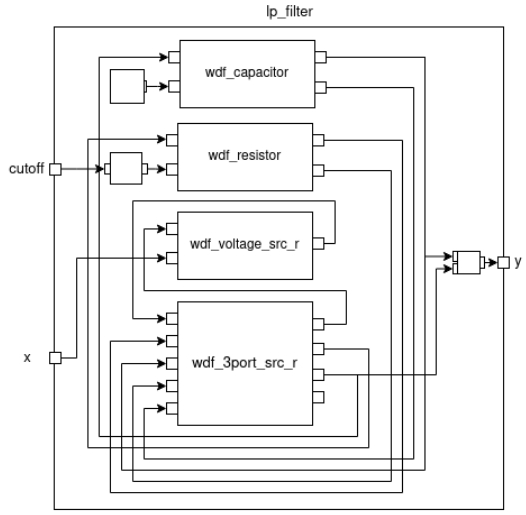


Figure 4: Simplified and unflattened graphical representation of the `lp_filter` WDF example.

Support for the implementation of wave digital models has been recently added to FAUST through the *WD-models* library [7]. It is based on an ad-hoc language extension referred as metaprogramming which uses a declarative style. While it manages to represent WDF systems and produce efficient code, we argue that it lacks modularity and flexibility due to the automatic inclusion of unit delays by the feedback operator ($\sim$). All necessary component-level delays are implemented at the tree level, in the most simple case by the `buildtree(connection_tree)` function, whose code reads [5]

```
builddown(connection_tree)~buildup(connection_tree) :
buildout(connection_tree)
```

In order for this to work properly, delays must be not implemented in the leaf component they would naturally belong to. For example, the capacitor component is implemented by the authors as

```
capacitor =
case{
        (0, R) => _*1;
        (1, R) => _;
        (2, R) => R0
        with {
                R0 = t/(2*R);
        };
}with{
        t = 1/ma.SR; //sampling interval
};
```

where clearly there is no delay between input and output, contrary to the standard WDF formulation.

[5] https://github.com/grame-cncm/faustlibraries/blob/master/wdmodels.lib

6. CONCLUSIONS

We presented the syntax and semantics of Ciaramella, a novel audio DSP programming language, and gave an overview of its compiler. Ciaramella combines a simple and declarative syntax with the SDF computational model to represent audio DSP systems. This choice results in an high level of modularity and composability which makes it possible to natively represent even complex systems such as WDFs.

The Ciaramella language and its compiler are however still at an early stage of development. Further work could include the implementation of conditional and loop blocks, multi-rate support, a library of common mathematical functions, n-delay operations, arrays and matrices, as well as more optimizations by symbolic simplification of expressions, pattern recognition techniques, etc.

7. REFERENCES

[1] "FAUST," https://faust.grame.fr/, accessed: 2022-02-08.

[2] "Max/gen," https://docs.cycling74.com/max7/vignettes/gen_topic, accessed: 2022-02-08.

[3] "Reaktor core," https://www.native-instruments.com/fileadmin/ni_media/downloads/manuals/REAKTOR_6_Building_in_Core_English_2015_11.pdf, accessed: 2022-02-08.

[4] "Soul," https://soul.dev/, accessed: 2022-02-08.

[5] V. Norilo, "Kronos: a declarative metaprogramming language for digital signal processing," *Computer Music Journal*, vol. 39, no. 4, pp. 30–48, 2015.

[6] S. D'Angelo, "Lightweight virtual analog modeling," in *Proceedings of the 22nd Colloquium on Music Informatics, Udine, Italy*, 2018, pp. 20–23.

[7] D. Roosenburg, E. Stine, R. Michon, and J. Chowdhury, "A wave digital filter modeling library for the FAUST programming language," 6 2021.

[8] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, 1987.

[9] K. Gilles, "The semantics of a simple language for parallel programming," vol. 74, 1974, pp. 471–475.

[10] E. A. Lee and D. G. Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," *IEEE Transactions on computers*, vol. 100, no. 1, pp. 24–35, 1987.

[11] S. Tripakis, D. Bui, M. Geilen, B. Rodiers, and E. A. Lee, "Compositionality in synchronous data flow: Modular code generation from hierarchical sdf graphs," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 3, pp. 1–26, 2013.

[12] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The synchronous data flow programming language lustre," *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1305–1320, 1991.

[13] T. Gautier, P. L. Guernic, and L. Besnard, "Signal: A declarative language for synchronous programming of real-time systems," in *Conference on Functional Programming Languages and Computer Architecture*. Springer, 1987, pp. 257–277.

[14] G. Berry and G. Gonthier, "The esterel synchronous programming language: Design, semantics, implementation," *Science of computer programming*, vol. 19, no. 2, pp. 87–152, 1992.

[15] N. Halbwachs, "A synchronous language at work: the story of lustre," in *Proceedings. Second ACM and IEEE International Conference on Formal Methods and Models for Co-Design, 2005. MEMOCODE'05.* IEEE, 2005, pp. 3–11.

[16] K. Barkati and P. Jouvelot, "Synchronous programming in audio processing: A lookup table oscillator case study," *ACM Computing Surveys (CSUR)*, vol. 46, no. 2, pp. 1–35, 2013.

[17] N. Halbwachs, "Synchronous programming of reactive systems," in *International Conference on Computer Aided Verification*. Springer, 1998, pp. 1–16.

[18] B. K. Rosen, M. N. Wegman, and F. K. Zadeck, "Global value numbers and redundant computations," in *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 1988, pp. 12–27.

[19] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[20] "dot - the fastest + concise javascript template engine for node.js and browsers." https://olado.github.io/, accessed: 2022-02-07.

[21] "Jison," https://github.com/zaach/jison, accessed: 2022-02-07.

[22] A. Fettweis, "Wave digital filters: Theory and practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, 1986.

Count-Me-In: A Collaborative Step Sequencer for Audience Participation

Juan Pablo Carrascal
Microsoft
jpcarrascal@acm.org

ABSTRACT

Modern musical events create a boundary between performers and the audience, with only the former playing and active role. We introduce Count-Me-In, a collaborative music sequencer that uses a distributed Web architecture to promote audience participation in music performances, installations, and other related contexts. Count-Me-In uses a step-sequencer design to reduce the negative effects of network latency on the participants' experience.

1. INTRODUCTION

In primitive societies, music has served as cohesive aspect of social activities such as religious rituals [1]. Music events are highly participatory, with all attendees playing an active role. In modern western society, where individual roles have become more and more specialized, participants of musical events are most often split into *performers* and *audience* [1] , with only the former playing an active role and producing music onstage.

Filipino musician and ethnomusicologist José Maceda criticized the commercial proclivity of western music [3], while studying indigenous music and advocating "*for a return to the experience of music as a social and communal event*" [4]. It is thus not surprising that some of his experimental works focused on using technology to bring the music performance back to a collective, participatory experience. In his work *Ugnayan*, he enabled large audiences to become performers by means of the *distributed speaker array* approach [5]: all of Manila radio stations broadcast the piece, while inhabitants were encouraged to go out with their portable radios to create a massive city soundscape. Similar participatory speaker arrays have become popular and used by several composers ever since. Please refer to Taylor's work [5] for a story of this approach.

A more contemporary take on participatory music makes use of modern computing devices like mobile phones to provide interaction capabilities to the audience [6–8]. An important aspect of this approach is to provide the audience with a clear understanding of the cause and effect relation between their actions and the resulting musical output. This aspect, also known as agency, is critical for the quality of musical interaction [9]. It has been observed that the perception of agency when observing a digital music instrument performance can be unclear [10]. Furthermore, providing agency to the *audience* while preserving overall musical structure, becomes a balancing act. Because of this trade-off, some systems are more suitable for ambient or experimental music. Solutions like voting systems [11, 12] have been used to achieve more structured musical results at the expense of agency [12].

Figure 1. Spontaneous Count-Me-In session with children.

In recent years, Web technologies have evolved to incorporate features that make it possible to develop real-time music applications. Schnell et al [13] make use of several modern Web standards to create a collaborative sound and visual installation. Matuszewski [14] created a framework for the development of distributed multimedia applications based on Web standards. Using Web technologies on mobile devices makes it extremely easy to deploy collaborative applications, as it relieves the need for installing dedicated applications, lowering the barrier for participation. Furthermore, the introduction of the Web MIDI API allows Web applications to communicate with external musical equipment, opening new possibilities for collaborative music performance.

With Count-Me-In, we aimed at creating a participatory and playful music experience that provides participants with agency while producing structured musical output. The system was designed around a collaborative step sequencer and leverages networked computing devices and modern Web technologies. Count-Me-In can easily fit spontaneous, 2-people sessions (e.g. Fig. 1), scenarios involving a larger number of people, or even combined with other electronic or acoustic instruments.

[1] Note that Small [2] considers many others, besides musicians and audience, to be participants of the "musicking" activity.

Copyright: © 2022 Juan Pablo Carrascal et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

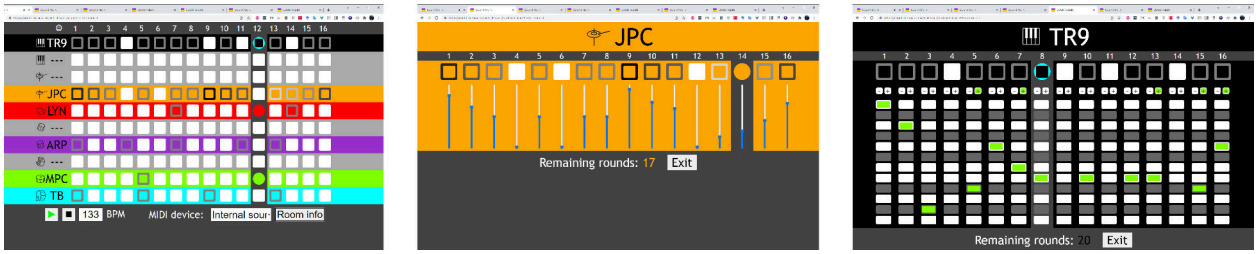


Figure 2. Count-Me-In Web apps. Left to right: the Sequencer, the Drum Track and the Synth Track.

2. DESIGN RATIONALE

Count-Me-In allows a group of people to become active performers and to create music together in a playful manner. We designed it with these principles in mind:

Collaborative. It engages audience members by making them active and conscious participants of the performance.

Social. It should promote social interaction. Focus should not be on each individual's interaction exclusively, but in the collaborative contribution. It should be a fun experience for all participants.

Interactive. In contrast to passive speaker arrays, members of the audience can control musical parameters. Causal relationship between their action and sound output should be clear (agency). Interactivity and agency allow members of the audience to effectively become performers.

Democratic. It should be very easy to use, regardless of age, technical knowledge, or musical proficiency. It should be platform independent, and it should not require a complicated installation or authentication.

Scalable. It should support a variety of scenarios with diverse numbers of people. It should work self-contained, but it should also be possible to use it along other electronic or acoustic instruments.

In alignement with these principles, we aimed for simplicity in sound and graphic design. We did not enable timbre editing or other complex functions and controls in the current iteration of Count-Me-In, as we wanted participants to focus on the collaborative experience. Some design decisions (e.g. visual cues like track color coding and labels) were aimed at helping participants to recognize their own and others' contribution to the musical activity, hence helping with agency and social awareness [15]. The effectiveness of these visual cues were evaluated in a preliminary study included in section 5.

3. SYSTEM DESIGN

3.1 Architecture

Figure 3 shows the system architecture. It comprises a back-end server running Node.js, a *Sequencer* Web application, and multiple mobile-friendly Web clients, the *Tracks*. The Web applications communicate in near-real-time with the back-end using WebSockets. Participant interaction data are logged to disk for analysis purposes. The Sequencer app runs on an orchestration computer, whose video output is meant to be shared with the audience via a large display or a projection screen. All system audio is rendered on the orchestration computer, hence, its audio output should be amplified.

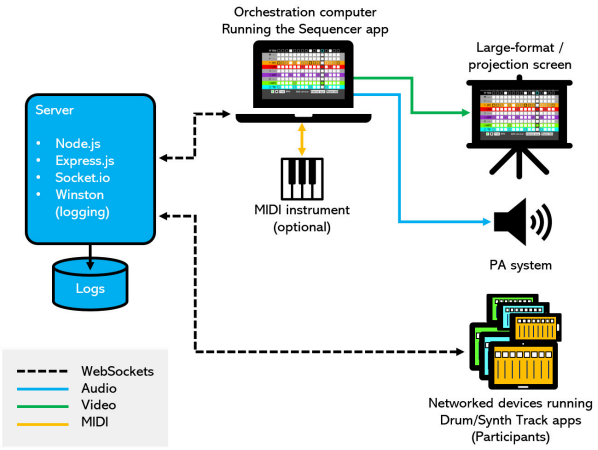


Figure 3. System architecture.

The system is designed around collaborative step sequencer. In a step sequencer, user actions (button presses) *schedule* notes to be played—as opposed to play them immediately. The sequencer then triggers audio at the right times, according to scheduled notes (steps). As user actions are not meant to trigger sounds in real time, the negative effects of network latency in participant engagement [16] are mitigated.

3.1.1 Sequencer and Track Apps

The Sequencer features 10 tracks and 16 steps and it is depicted in Figure 2 (left). It is worth noting that the number of tracks is not hard-limited to 10. We used a number that allowed all tracks to be comfortable shown on a typical 16:9 display. However, more tracks could be fit using a large vertical display. Adding new tracks only requires trivial code adjustments. Sequencer controls include Play and Stop buttons, Tempo value in beats per minute, a selector to switch between internal or external sounds, and a button to display the current *session* information. As the tempo is determined by the Sequencer app, master tempo adjustments should be done in the orchestration computer. We plan to implement a feature to overcome this limitation in a future iteration of the Count-Me-In.

The Drum and Synth Track applications are shown in Figure 2 (middle and right, respectively). The Drum track features 16 switches to turn individual steps on or off, and

16 slider controls to adjust the velocity of each step. The Synth Track features a 1-octave keyboard per step centered at the C2-B2 octave (C2 = 65.41Hz), with switches to transpose an octave up or down.

3.1.2 Sound Generation

We implemented a simple square-wave-based monophonic synthesizer and a 8-track drum sample player using the WebAudio API [17]. The synth was based on a simple square wave, with 30 milliseconds of portamento. Drum samples were a subset of the popular Roland TR-808 [2] sounds (Kick Drum, Snare Drum, Clap, Hi Tom, Mid Tom, Cowbell, Closed Hi-Hat, and Open Hi-Hat). Two instances of the synthesizer are assigned to tracks 9 and 10 respectively, while different drum sounds are assigned to tracks 1 to 8. Further sound design for the system is possible but out of the scope of this paper. However, the Sequencer output can also be sent to an external MIDI device if external sounds are preferred. The use of MIDI also makes it possible to synchronize Count-Me-In with other electronic instruments. MIDI implementation was done using the Web-MIDI API [18].

3.2 Session Workflow

Figure 4 depicts the workflow of a Count-Me-In *session*, as explained next:

❶ The Orchestration computer should first access the Count-Me-In server [3] and provide a name for the session. ❷ The computer will display the Sequencer app. Its tracks will all be empty and greyed out. A modal window (❸) containing session information will be initially overlaid over the Sequencer. It includes the name of the session, a URL for participants to join the session, and a QR code. At that moment, the session has started. ❹ Once the session information modal is closed, playback will begin. Playback can also be manually started and stopped, and tempo can be adjusted with the relevant controls (Figure 2, left).

❺ Multiple participants can join the session by scanning the QR code or entering the URL in a Web browser. ❻ They are asked for their initials, and they can click a button (labelled "Go!") to enter the session. When a participant enters a session(❼), they are randomly assigned one of the sequencer tracks. ❽ The Track app (either Drum or Synth) will be displayed on their device. Besides the track controls, Track apps display an icon that represents the instrument controlled by the participant and the participant's initials. Each Track has a distinct color, depending on the assigned instrument (Figure 2, center and right). When a participant joins, the Sequencer app will highlight the corresponding app with a color that matches that of the participant's Track, and will display the participant's initials at the left side of the Sequencer track (note the matching colors and initials in Figure 2). We hypothesized that with this the combination of visual cues (color-coded tracks, initials, icons), each participant can easily notice what Sequencer track they are controlling, and hence understand how their interactions result in changes in the sound output.

❾ As soon as a participant starts interacting with the Track app (e.g. by clicking a step button), 20 loop iterations will be counted. After the 20 iterations have passed, the participant will lose control of the track and will be dropped in the initial screen (their initials are remembered by means of a client cookie). The track is then released for use by other participants, but the changes previously made to the Track remain so as not to disrupt the current composition being played. This turn-taking mechanism prevents participants to take exclusive controls of a track for a long period of time. This also allows a number people higher than the number of Sequencer tracks to participate in the session. We thought it made more sense to set this timeout in musical terms, hence we chose a number of iterations instead of a period of time. 20 loop iterations (40 seconds at 120bpm) seemed adequate in our tests, but further evaluation with participants might be necessary to find the sweet spot. Also, there is no reason for this to be a set number, it could be adjusted depending on the Sequencer tempo or the number of participants.

4. PRELIMINARY EVALUATION

We conducted a simple evaluation study with Count-Me-In. This was not meant to be a thorough evaluation of the system, but a way to obtain preliminary feedback to validate some of our design decisions. Specifically, we wanted to determine whether the combination of visual cues (track color, initials, icons), and simple interface controls used in the design of the Sequencer and Track apps, allowed participants to understand the cause and effect relationship between their individual actions and the resulting collaborative output.

We invited 5 people with ages ranging from 8 to 46 years (mean 34.4), 4 of them female, to participate in the evaluation. None of the participants had a background in music, although the youngest (an 8-year old child) has started his musical training. The participants were attending a small social gathering. Given the social aspect of Count-Me-In, we consider this was an adequate context for evaluation.

We set up a computer with a projector and a speaker, started a Count-Me-In session and invited participants to scan the QR code, enter their initials and join the session. We were intentionally brief and vague when providing instructions to participants. We simply asked them to try to use the Web app on their mobiles (we provided an 11-inch tablet to the child) and to observe what happened in the projected app. We then observed and took note of participants' reactions and behaviors.

To avoid interrupting the social activity, we did not ask participants any follow-up questions immediately after the Count-Me-In session took place. Instead, we asked participants what they liked and what they disliked about the session the following day, via text messaging. When needed, we also asked some clarifying follow-up questions, including some regarding the Drum and Synth Track user interfaces (UI). It remains part of future work to conduct a more in-depth study that might include a post-session focus group or individual interviews.

[2] https://en.wikipedia.org/wiki/Roland_TR-808

[3] Currently deployed at https://count-me-in.azurewebsites.net

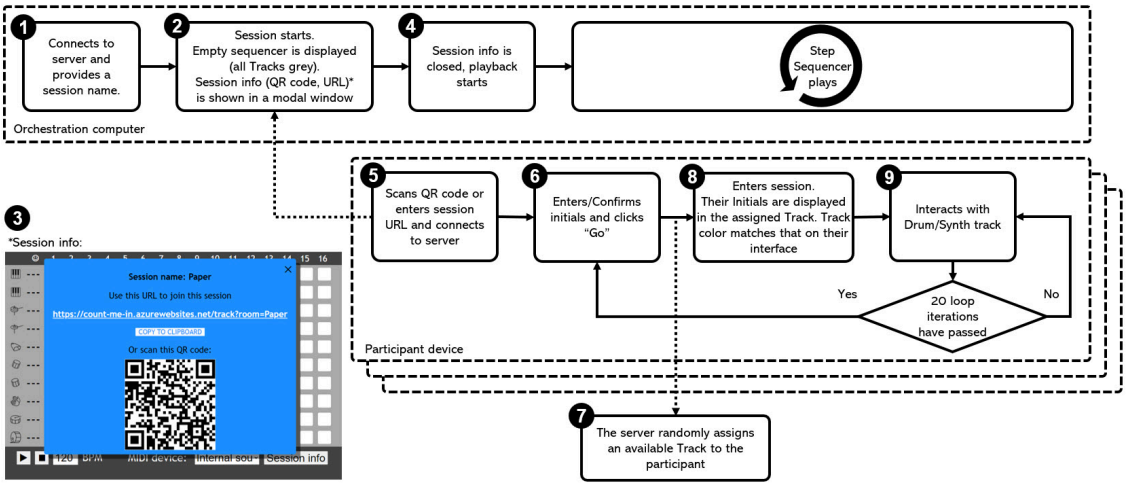


Figure 4. Session workflow and session information window (lower left corner).

Figure 5. Evaluation study setting.

5. RESULTS

We logged activity information in the server during the almost 17 minutes that the session lasted. Our logs were time-stamped and included connection and disconnection of each participant as well as interaction events. Interactions could be button clicks (either on the Drum Track or the Synth Track) or slider movements (only available in the Drum Track). During the session, the 5 participants produced a total of 2018 interaction events (min: 237, max: 620, mean: 403.6), for an average of 23.7 events per minute per participant. While there is no baseline to compare with, this is evidence that participants were engaged in the activity.

Overall, the reception to the Count-Me-In session was positive. Based on the way people interacted with the system and with each other, we observed two phases in the activity. In the first phase, as soon as the activity started, participants experimented with the interface, clicking buttons and tweaking controls to see what happened. Phase two came after a few minutes of experimentation, when participants started to have a more strategic approach, teaming up to have structured results. Post-hoc discussion with P2 corroborated the existence of these 2 phases. She explained: *"What I liked most about the instrument was the social element and the way we could explore and make sense of it together, both individually in our own devices, and as a group, on the wall projection and in our discussion. At one time, someone said, let's work together, and I tried coordinating more with the others and creating some sort of visual harmony that I hoped would translate into musical harmony"*. P4 also appreciated the collaborative possibilities of the system and observed that *"[it is] an activity that is shared among several people to form a team together"*. The experience of "making music" with Count-Me-In was generally considered fun and playful, and seemed to elicit engagement and social connection between participants: *"It was fun to be able to interact with a melody and with others"* (P5). A most concise piece of feedback came from P3, an 8-year child, who simply said *"I liked it all"*. We tried to obtain more critical opinions from him, to no avail.

5.1 Effectiveness of Visual Cues

Our observations during the session and subsequent responses to post-hoc questions indicated that the visual cues provided participants a sense of agency. After the initial instructions were given and they had joined the session, participants did not ask for additional guidance from the researchers. During the activity, we asked participants if they understood which Sequencer Track was theirs, and the answer was affirmative in all cases.

Visual cues, specially color coding, seemed to be effective in guiding participants through their interaction. Colors were also considered a pleasant aspect of the interface. As explained by P1 *"The interface on the screen was pretty, and the little colors were easy to understand"*. When we asked P4 what helped her the most to identify what sound she was in control of, she stated *"with the button pushing and the assigned icon"*, indicating icons were indeed a helpful visual cue for participants. However, P2 also observed that the icon used in the Synth Track (a generic 1/2 octave keyboard image) misled her to expect the sound of an actual piano.

Participants' answers to post-hoc questions did not ref-

erence the use of the initials as Track identifiers, but we observed a lot of lively conversations about who was in control of each of the active Tracks. Initials were therefore used to recognize what Tracks *others* were in charge of, as opposed to which one was their own Track. Understanding the contributions of peers by reading each other's initials helped developing the observed collaborative strategies after the initial exploration phase.

5.2 Sound Recognition by Participants

While the visual cues seem to be helpful for participants to identify their Tracks, recognizability of sounds varied between instruments. P1 indicated that *"Some instruments could be heard or were more obvious then others, which made it difficult to recognize what one was doing"*, In P2's words, *"Some of the most difficult to understand and feel like I was indeed controlling them, were the pianos and some drums. I couldn't tell what kind of drums they were, so it was difficult to hear myself, but others, like the clapping, were easy to identify."* P4 even suggested to *"use a single earphone per person to better listen each one's instrument"*. A more extreme case, P5, felt she could not never find her sounds, and explained that *"I could find myself clearly in the projection [but] I would have liked to be able to find [the sound of] my instruments"*. We hypothesize the difficulty to recognize some of the timbres had to do with the sub-optimal quality of the sound amplification system used during the session (the speakers in the projector), as we observed sounds like the kick drum were particularly difficult to hear.

5.3 Feedback on the Drum and Synth Track UI

Two out of 5 participants considered the Synth Track interface difficult to use. In P1's words, *"I didn't understand the more complex interface of the keyboard"*. We used a vertical button strip with black and white keys color coding (see Figure 2, left). Users with musical training would probably notice the black and white pattern and vertical orientation commonly used in piano rolls but, admittedly, it does not look like a real keyboard. In contrast, we observed that participants used the Drum Track interface without noticeable issues or complaints.

We did not hear any criticism regarding the Drum Track during the activity, and we observed that all participants seemed to quickly understand how to use it. However, most of the interaction events were button presses (1379, 68%), with less than 1/3 being slider adjustments. To understand why, we asked participants what they thought the sliders were for, and we discovered that they were not well understood. P4 said *"I have no idea. I presume for turning up the intensity"*, while P1 stated *"I think it was the volume"*. The remaining participants had other assumptions, but nobody seemed to clearly understand the function of the sliders.

Finally, there was some confusion among participants regarding the notes that were already set in the Track when they joined or re-joined the session after their turn expired (see ❾ in section 3.1.2). They thought they were placed there as an example, or simply did not know where they came from. This was not problematic, but it was confusing, nonetheless.

In the next section we elaborate on the results and propose potential solutions for some of the issues we identified.

6. DISCUSSION

The results of the preliminary evaluation indicate that Count-Me-In has potential for creating playful musical engagement in social contexts. Visual output, specially color coding and icons, effectively helped participants to understand their contributions to the music activity. Initials identifiers were more frequently used by identifying the contributions of *other* participants, which was an unexpected use of the feature. We think this obeyed the need for social awareness, which is a relevant aspect of communal music activities. As explained by Professor Iain Morley, when discussing what constitutes participation in musical and ritual performances, *"to consider oneself a participant [...] in a performance (ritual or musical) relies upon a strong sense of the perception of you by others engaged in the activity, that is, it relies on well-developed theory of mind and social awareness"* [15]. Being able to identify each other's contributions encouraged participants to develop collaborative strategies during the music making activity.

The aesthetics of the Sequencer interface were appreciated by participants, suggesting that the session provided a rich sensory experience: *"I tried coordinating more with the others and creating some sort of visual harmony that I hoped would translate into musical harmony"* (P2).

There is still work to be done regarding the Track interfaces. The vertical piano roll in the Synth Track was difficult to understand. We will consider other approaches. One could be to redesign it to be more piano keyboard-like. Another approach might be to use a slider to control pitch. Unfortunately, sliders were not easily understood by participants in the Drum Track, where they were used to adjust the velocity of each step. Participants indeed explored the sliders, but presumably did not understand the connection between the slider position and the velocity of the drum sounds. We will need to further explore whether this lack of clarity remains when the sliders control pitch, as opposed to velocity. As for the Drum Track, possible improvements include adding labels to the sliders or replacing the sliders altogether with radio buttons with different levels.

We did not receive any critical feedback that could be related to the responsiveness of the system. Further technical evaluation is necessary, but our preliminary study indicates that the step-sequencer design successfully made the effects of network latency and jitter irrelevant to the participant engagement.

We conclude that the combination of visual cues and a step-sequencer design generally helped providing agency to participants. However, this result was not consistent in all circumstances. Participants found it difficult o recognize some of the sounds used by Count-Me-In. Low-frequency sounds, like the kick drum and the center octave used by the Synth Track (C2 = 65.41Hz) were harder to

hear. The latter case was aggravated by using an interface that was hard to understand by our non-musically-oriented participants. We think that the use of a low-quality amplification system had an important influence in this result, as low frequencies were not reproduced in an adequate manner. Further evaluation with a proper, full-range amplification system will help clarifying this. Using a higher center octave for the Synth Track would also help users to recognize its sound.

6.1 Future Work

The initial results of the Count-Me-In are promising, but there are areas of improvement. In future work we will include the evaluation of improvements mentioned in the precious section; understanding the influence of sound quality in the perception of the different sounds; and a study with focused on verifying that the turn-taking system allows larger number of users to engage with the system.

We also envision different scenarios in which Count-Me-In can play an important role in engaging attendees. First, we are interested to see how current results vary with expert subjects, thus we plan to evaluate the system with users with musical training. We hypothesize that there will be less issues with the interface and sound recognition, and we think the input of musicians might be valuable on other aspects. Additionally, we want to evaluate the value of using the system along with live performances of other instruments. In that scenario, we think Count-Me-In will help creating a cohesive, engaging and fun experience involving both performers and the audience.

7. CONCLUSION

We introduced Count-Me-In, a collaborative music making and performance system. We conducted a preliminary evaluation and found evidence that it helps creating a fun, playful and engaging experience for social contexts, satisfying most of our design principles. We identified shortcomings of the interface and observed that the experience of individual users might be affected by the timber of the instrument they are controlling. We also observed that the sound quality of the amplification system is important for an optimal experience. We plan to refine the system based on the feedback received in the preliminary study, and to evaluate it in different contexts, including larger user groups and sessions with musicians.

8. REFERENCES

[1] B. Nettl, *Music in primitive culture*. Harvard University Press, 2013.

[2] C. Small, *Musicking: The meanings of performing and listening*. Wesleyan University Press, 1998.

[3] J. Maceda, "A primitive and a modern technology in music," *Unpublished typescript. University of the Philippines*, 1978.

[4] E. Bentcheva, "From ephemeral experiences to lasting legacies: Discourses on experimental art in the philippines during the 1960s and 1970s," *Tate Papers*, vol. 32, 2019.

[5] B. Taylor, "A history of the audience as a speaker array." in *NIME 2017*, 2017, pp. 481–486.

[6] N. Weitzner, J. Freeman, S. Garrett, and Y.-L. Chen, "massmobile-an audience participation framework." in *NIME 2012*, vol. 12, 2012, pp. 21–23.

[7] A. Hindle, "Swarmed: Captive portals, mobile devices, and audience participation in multi-user music performance." in *NIME 2013*, 2013, pp. 174–179.

[8] J. J. Arango and D. M. Giraldo, "The smartphone ensemble. exploring mobile computer mediation in collaborative musical performance." in *NIME 2016*, 2016, pp. 61–64.

[9] O. Bown, A. Eldridge, and J. McCormack, "Understanding interaction in contemporary digital music: From instruments to behavioural objects," *Organised Sound*, vol. 14, no. 2, pp. 188–196, 2009.

[10] F. Berthaut, D. Coyle, J. W. Moore, and H. Limerick, "Liveness through the lens of agency and causality," in *The 15th International Conference on New Interfaces for Musical Expression, 31 May-3 June, 2015, Louisiana State University, USA*, 2015.

[11] M. Gimenes, P.-E. Largeron, and E. R. Miranda, "Frontiers: Expanding musical imagination with audience participation," in *NIME 2016*. Brisbrane, Australia, 2016.

[12] L. Zhang, Y. Wu, M. Barthet *et al.*, "A web application for audience participation in live music performance: The open symphony use case," in *NIME 2016*, 2016.

[13] N. Schnell, B. Matuszewski, J.-P. Lambert, S. Robaszkiewicz, O. Mubarak, D. Cunin, S. Bianchini, X. Boissarie, and G. Cieslik, "Collective loops: multimodal interactions through co-located mobile devices and synchronized audiovisual rendering based on web standards," in *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction*, 2017, pp. 217–224.

[14] B. Matuszewski, "A web-based framework for distributed music system research and creation," *Journal of the Audio Engineering Society*, vol. 68, no. 10, pp. 717–726, 2020.

[15] I. Morley, "Music and ritual: Parallels and practice, and the upper palaeolithic," 2009.

[16] J. Lazzaro and J. Wawrzynek, "A case for network musical performance," in *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, 2001, pp. 157–166.

[17] "Web audio api," https://www.w3.org/TR/webaudio/, (Accessed on 01/27/2022).

[18] "Web midi api," https://www.w3.org/TR/webmidi/, (Accessed on 01/27/2022).

DESIGNING THE BALANCE BETWEEN SOUND AND TOUCH: METHODS FOR MULTIMODAL COMPOSITION

Claire Richards,[abc] *Roland Cahen,*[b] *and Nicolas Misdariis*[c]
[a]Actronika SAS
[b]Le Centre de Recherche en Design ENSCI Les Ateliers/ENS Paris-Saclay
[c]STMS-IRCAM-CNRS-SU-Ministère de la Culture
clairicha@gmail.com, roland.cahen@ensci.com, misdariis@ircam.fr

ABSTRACT

In this paper, we discuss key research questions generated from a collaborative workshop, during which our aim was to explore the potential of a wearable device to produce novel audio-haptic sensory experiences. The main intention of this research is to enable users with any type of hearing profile to appreciate a body-centered listening experience. The multimodal harness, developed by the co-authors, integrates nine voice coil actuators into a wearable structure, stimulating both the auditory and tactile senses via extra-tympanic sound conduction and vibrotactile stimulation of the skin on the upper body (spine, clavicles, ribs). We need to create our own interfaces in order to judge its capacity to elicit three modes of vibratory sound perception: auditory, tactile and bi-modal. To this end, we used the Max environment to make several preliminary authoring tools, whose compositional features allow us to explore three main themes of sensory composition: multimodal music listening experiences, spatialization of audio-haptic signals on the body, and sensory equalization. Feedback from trial sessions, along with current constraints due to the wearable and interface design, give us direction for our future work: iterate to improve the sensory experience of the multimodal harness, and apply these tools experimentally in order to contribute to multisensory processing research.

1. INTRODUCTION

The experiences of grazing one's hand across a soft surface and listening to a concerto may not seem to have much in common. However, in both cases, the body is detecting and interpreting vibratory information, whether that vibration is a result of the friction produced by the movement of skin on a surface, by the movement of performers' bows on the strings of violins, or by touching the surface of a speaker playing music. In our research, we seek to illuminate the physical aspect of sound perception, by bringing the vibrations directly in contact with the body. In parallel to the development of our multimodal wearable device, we conducted a psychophysical study on extra-tympanic hearing thresholds of the human torso, with the aim to begin a definition of audio-haptic signal design guidelines for the wearable [1]. In this study, instead of taking a psychophysical approach, we decided to start by applying the basic parameters obtained from the first study's results within a more creative developmental framework: an intensive workshop.

Our intention in organizing this workshop was not to produce data, but to gain more clarity on the research questions we can ask. In the next sections, we first present the project context of exploring a set of audio-haptic forms of expression, followed by basic technical details of the multimodal harness. Then, we describe functionalities of authoring tools we developed, trial feedback, and the research questions generated by the sensory experience they permit (or not). Finally, we discuss our perspectives on future development, for both experimental and compositional purposes.

2. CONTEXT AND MOTIVATIONS

The perceptual ranges and discriminatory limits of hearing and touch are compatible and overlapping. Understanding the body's ability to interpret vibrotactile information can be broken down into several key characteristics of the presented stimulus: its frequency, intensity, duration, waveform and position on the skin's surface. The elements of vibrotactile signal design have much in common with those of auditory signal design (pitch, amplitude, duration, timbre, distance), but the two senses do not process the sensory information in the same way [2]. Opposed to hearing, which is centralized to the head, we perceive vibrations all over our body [3]. The perception of vibrations in the frequency range between approximately 10-1000 Hz overlaps the detection capacities of the two senses.

Though there are distinct differences in processing capacities between the two sensory organs, researchers have found that auditory stimulation can affect tactile perception [4, 5], and vice versa [6, 7]. Beyond influencing the perception of presented auditory stimuli, tactile stimuli alone have been found to activate the auditory cortex, both in normal-hearing and congenitally deaf adults. Researchers have found this effect to be greater for deaf versus hearing participants [8]. The mutual facilitation of one sense on the perception of the other can be attributed to

processes of multimodal integration and cross-modal enhancement [9].

The perception of hearing and touch can therefore be influenced separately and simultaneously, by using the same source of stimulation: vibration. In this paper, we discuss the excitation of the two modalities by means of extra-tympanic conduction of sound and vibrotactile stimulation of the skin. The existing literature on extra-tympanic conduction provides an essential foundation towards understanding how we are able to hear sounds that do not stimulate the outer ear. However, notable past investigations of these mechanisms are primarily based on the anatomy of the skull [10]. Commonly described as bone conduction, this paper refers to contact-based sound transmission as "extra-tympanic", because there is speculation that soft tissue is involved in transmission mechanisms, and researchers have found that sound can be conducted from other parts of the body [11].

Many researchers have therefore investigated the use of vibration technology to augment auditory experiences by creating interfaces for the body. A large portion of these interfaces were specifically created for the improvement of musical experiences, and within this category, many devices have been developed specifically for the back and upper body. The model human cochlea [12] is a tactile device that separates audio signals into frequency bands from 20-1000 Hz, displayed to the back via 16 voice-coil actuators, arranged on the back of an armchair. The Haptbeat and Vibeat both integrate motors that vibrate in time with music, worn as a necklace or bracelet. The Soundshirt integrates 30 micro-actuators (type unspecified) in a thin, flexible garment created for hearing-impaired individuals to have a tactile experience of philharmonic music. Users feel lower frequencies in the lower abdomen, while higher frequencies are displayed across the upper body. This frequency-based organization of tactile stimulations draws comparison to the model human cochlea while it differs from the Subpac, a backpack-style vest that enhances bass frequencies via two large voice coil actuators located on the back.

Our research differentiates itself from the prior art through our approach to mixed-modality effect design. The resulting experience of sound, perceived either by means of extra-tympanic hearing, tactile stimulation of the skin or a mixture of both modalities, is accessible to all hearing profiles. Hearing loss comes in many shapes and sizes, differing based on the frequencies affected, severity, symmetry, and type of lesion (conductive, sensorineural or mixed) [13]. Someone with severe sensorineural hearing loss could appreciate tactile sensations, while an individual with conductive hearing loss could hear the wide range of acoustic frequencies displayed via extra-tympanic conduction. Beyond catering to diverse hearing profiles, our project addresses normal hearing profiles as well, and is inspired by the principles of universal design: conception whose aim is to include all users, regardless of ability.

Figure 1. Image of the multimodal harness. There are nine actuators integrated on the structure: two on the clavicles, two on the ribs, and five along the spine. The straps are velcro-adjustable, two of which go around the legs to secure the last module against the body. The multichannel audio-haptics card is connected to the harness, situated below the last module on the spine.

3. THE MULTIMODAL HARNESS

3.1. Conception and Functionality

Before beginning the design process [1], we defined two key constraints for the multimodal harness: adhesion of the module against the body, and user comfort. In order to achieve effective sound transmission and avoid signal attenuation, the vibrating module should be made of hard material, secured tightly against the body. Balancing tightness and comfort, two contradictory qualities, became a principal design challenge which we addressed with the flexibility of the modules and the ergonomic aspects of the harness structure (fig. 1).

The technical aspects of the module (fig. 2) allow them to adapt to the body's form at the different points of stimulation (spine, clavicles, and ribs). The hard plastic parts in direct contact with the user are ergonomically shaped (wide and flat on the ribs, extruded on the lower spine, rounded on the clavicles) in order to maximize the ratio of comfort to auditory signal transmission. The harness structure (fig. 1) has adjustable Velcro straps attached to all corners of the modules, pulling them against the body. Nine of Actronika's proprietary voice coil actuators are connected to the Mk-1 20-channel audio-haptic card. The Max environment interfaces with the USB Audio output on the Mk-1 board, which is detected as a sound card.

[1] A co-development organized with Actronika, industrial and research partner, and Les Vertugadins, a Parisian costume design studio.

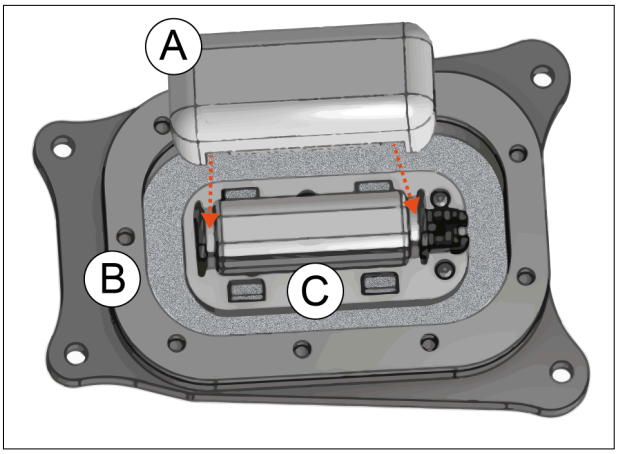


Figure 2. Schema of the module, which integrates the actuator. The module snaps onto extended fabric so it can "breathe" up and down on its frame (B), attached via the PCB clip (C). The 3D-printed clip-on piece (A) attaches onto the indented edges of the actuator, and comes into contact with the body, transmitting the vibrations to the user.

Each of the nine actuators on the multimodal harness has its own channel on the multichannel audio-haptic card. In its current wired design, users' movements are restricted to the length of the 9V power supply and audio signal transmission cables.

4. DESIGN WORKSHOP

The two-week workshop, organized by the co-authors, took place at IRCAM in Paris, France in August 2021. During the workshop, five participants (P.01, P.02, etc.) tested our tools based on their own interest or direct involvement in the project: they were neither specifically selected nor remunerated for their participation. During every trial of the multimodal harness, users wear ear plugs and a noise-canceling headset in order to mask the external noise created by the modules' vibrations and focus on their hearing via extra-tympanic conduction.

The main objective of the design workshop was to create first drafts of audio-haptic signal design interfaces. Creating these tools would allow us to refine our assumptions about the sensory experiences afforded by the multimodal harness, informed by our previous psychophysical study, haptic illusions of signal movement [14], and existing research on vibrotactile musical experiences [2, 12].

We hypothesize that the vibratory activation of tactile, auditory and bi-modal perception enables myriad possibilities for signal design: sensations that can travel in all directions to create illusions of movement, sound perception by vibratory transmission through musculoskeletal structures, and novel effects based on the multimodal integration of both sensations. While it is possible to approximately localize tactile stimuli on the body's surface [15], auditory perception by means of extra-tympanic conduction on the torso is essentially mono, since all sounds are entering the inner ear from the same pathway: the cervical spine. In other words, spatial audio information is lost, but we hypothesize that the localization of tactile stimuli could remain strong enough to transmit impressions of spatialized auditory source positions.

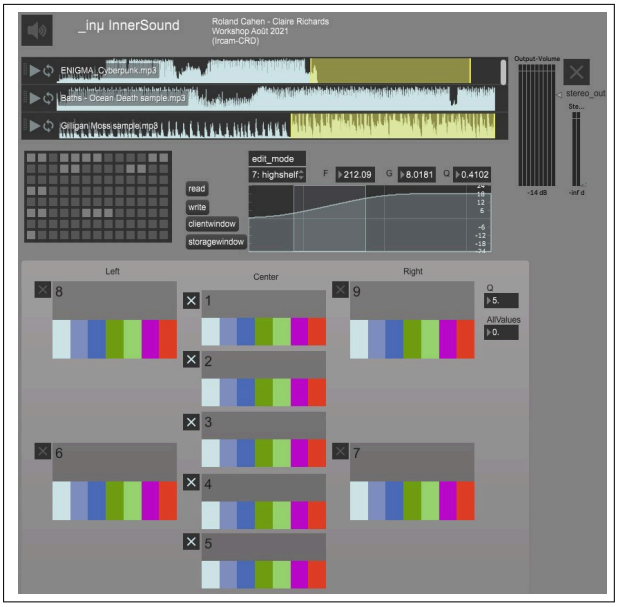


Figure 3. Inner.Music as seen in Max environment. Any sound file can be uploaded into the playlist object at the top of the interface. By default we propose a short list of rhythmic music with powerful bass content: electronic, disco-funk and pop. Left and right channels are routed to the left and right lateral actuators on the clavicles and ribs and mixed together on the vertical positions of the spine. A graphic equalizer and a mute button is available for each actuator, with different colored bars for 7 frequency bands spanning 50-4000 Hz. The user can save their equalizer settings in the grid below the playlist object, recorded and saved for future use. To the right of the preset grid, a high shelf filter object, applied to all channels, attenuates the bass frequencies with respect to the high frequencies due to the resonant frequency of the actuator (70 Hz).

5. AUTHORING TOOLS FOR THE MULTIMODAL HARNESS

Each of the following authoring tools offers a different possibility for designing audio-haptic effects. In this section, we describe each of the authoring tools in three parts: the interface design, feedback from trial sessions, and finally key questions and limitations related to interface functionalities and perceptive experiences.

5.1. Inner.Music: Multimodal Music Player

Inner.Music is a music player, designed to test the perception of filtered source files transmitted to the multimodal

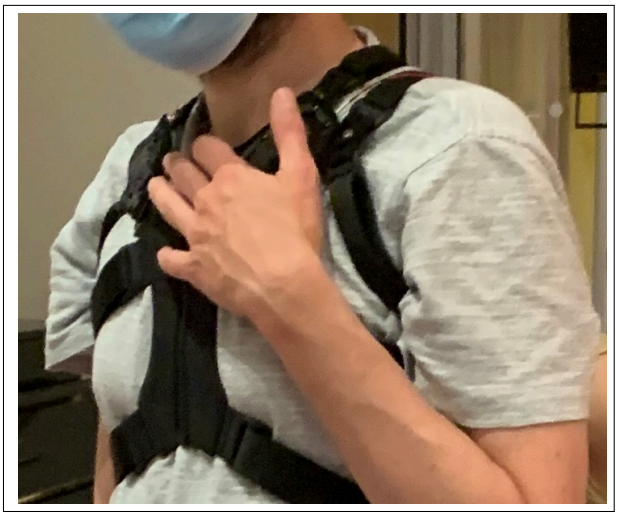


Figure 4. Photo of P.01, taken during the workshop. Since he wasn't satisfied with the quality of sound perception, he placed his fingers directly on the modules to enhance his tactile perception.

harness. This tool features a body point-based equalizer tool, which allows the designer to calibrate the signals to each position of stimulation.

5.1.1. Interface Design and Composition Features

Inner.Music allows the designer to self set, adjust and balance the volumes of each actuator's filter bands, while listening to music. The interface is also a preset manager: individual sensory profiles, created by saving the nine filter band settings in presets, can vary each person's sensory preferences (see fig. 3).

Our initial intention for Inner.Music was to create a tool that adjusts each actuator's spectrum in order to obtain a homogeneous audio-haptic sensation across all stimulated positions. However, we understood during the trial sessions that the ranges and nature of the two modalities' perceptions vary across positions, and according to different types of effect design or scenarios of use. We therefore chose to exploit these variations, creating an interface that allows the user to tweak their own sensory experience instead of imposing a generic predetermined sensory calibration.

5.1.2. Trial Feedback

While toying with the Inner.Music interface, participants employed various strategies: balancing the signal to achieve a perception of homogeneity across the positions, attributing levels and spectrum bands to each position in order to focus their perception on either the tactile or auditory modality, or muting positions that they did not appreciate. For example, some favored a stronger tactile sensation of the bass in the lower back, and routed the trebles to the upper back and clavicles to hear via extra-tympanic conduction. A common remark was that the music is audible, but appears to be coming from a distant place within their body, and in order to hear more details, they have to focus their attention. P.03 said, "It's possible to hear the sound, but to feel practically nothing." P.05 was especially sensitive to extra-tympanic auditory perception, saying: "I can hear the sound as if I were listening with headphones - maybe even better". On the other hand, P.01 was disappointed by the quality of the auditory experience and resorted to actively touching the modules to enhance his tactile perception (fig. 4). These differences in sound perception could be attributed to morphological differences affecting the transmission of the mechanical waves within the body [11]. A user's expectations may also impact how they judge their sensory experience: they might compare the quality of their auditory experience to other systems that were specifically designed for the ears, not the body.

5.1.3. Key Questions and Limitations

The strong morphological differences between each individual can be compared in a similar way to how differences in the structure of the outer ear can affect one's hearing. The Head Related Transfer Function (HRTF) is a physical transfer function that describes how a given sound coming from a specific point will reach the ear, in terms of spectral characteristics [16]. In comparison, we wonder if we could develop a Body Related Transfer Function (BRTF), a functionality capable of adjusting the multimodal harness's response to each individual in order for them to perceive the same signal, based on their morphological differences (height, weight, musculoskeletal characteristics).

5.2. Mp2p: Body Spatialization Interface

We created the Mp2p (Mono point-to-point) interface as a creative tool for spatialization of multimodal signals on the harness, by manipulating the signals' positions and trajectories.

5.2.1. Interface Design and Composition Features

Within the design space, simple mono signals can be played on any single actuator or grouped set of actuators and panned [2] between them according to a set 1D path. Spatialization of the multimodal signals in the Mp2p system works based on configuration presets which organize the points of stimulation along different paths. The signal can be routed to each independent point, one channel after another, or to several points at once, by grouping channel outputs together. In fig. 5, we see three possible configuration presets: these allow the designer to create different illusions of movement (sporadic in the first preset, side-to-side in the middle preset, top-to-bottom in the right preset). The actuators' positions on the lateral and vertical axes of the body allow for directional and alternating effects. We defined a set of configurations, providing a

[2] Linear panning with Max [mc.mixdown @pancontrolmode 2] object gave the best results.

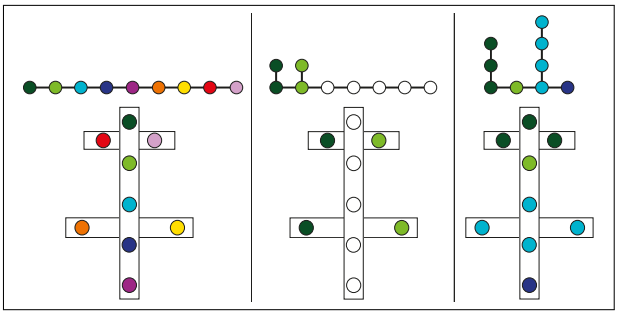


Figure 5. Diagrams of three possible Mp2p spatialization path configurations. On the top of the figure, arrays of colored dots represent the signal path: each different color is a different channel. Below each signal path is a graphic representation of the nine positions on the harness structure: the upper horizontal line is the clavicles, the lower is the ribs, and the vertical line is the spine. In the left-most configuration, each position has its own independent channel. The middle configuration uses only two channels (light and dark green), with two positions paired to each channel - the five spine points are inactive. The right-most configuration groups the positions among four channels: the clavicles and top spine point (dark green), the second spine point (light green), the middle spine and ribs (cyan) and the lowest spine point (navy).

diverse selection of routing structures for the spatialized effects. The designer can manipulate the signal's position, duration, waveform (sine, saw tooth or noise), frequency (50 Hz to 4 kHz), and linear amplitude (0. to 1. float multiplier), and play their composition looped or one-shot (see fig. 6). By manipulating these variables, the sensory effects can be perceived as having continuous or discrete inter-position transitions: the signal can move gradually, or switch abruptly along its path.

5.2.2. Trial Feedback

The Mp2p interface is slightly more complex than the other two tools, so after learning about the basic parameters, participants generally chose to tweak our predefined patterns rather than create their own from scratch. P.03 was particularly impressed by the sensation created by the signal moving up and down the spine. If the signal moved quickly enough along the points, he felt a continuous sensation, like a "hand running up and down [his] back." The slower the movement of the signal along the points, the easier it was to localize from which position it came. The illusion we tried to create of a front-to-back shifting sensation between the front actuators (clavicles and ribs) and back (spine) was less convincing, perhaps due to the slightly central position of the actuators on the ribs, and because the sensation of something crossing one's body is a rather alien concept. The duration of the experience as a whole also impacted the sensory experience: after trying the patterns, P.02 said:

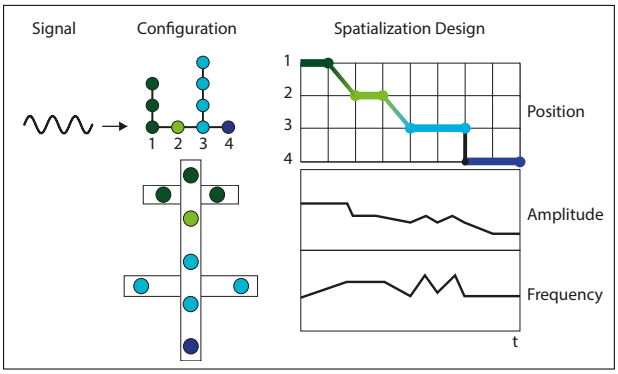


Figure 6. Graphic representation of the Mp2p interface functionality. A sine-wave signal follows a 1D path according to the chosen configuration preset (see fig. 5). To the right of the figure is the spatialization design, with three curves for position, frequency and amplitude. By modifying the curve, the designer can fine-tune the signal's duration at each group of positions, and its variations in amplitude and frequency across the composition.

I need more time to become familiar with the sensations I'm feeling on and in my body, since they are so new. Just a few minutes is not enough for my brain to really understand what is happening. I think with more time I'd be able to appreciate the sensations a bit more.

Participants made more comments on their "outer" body (tactile perception) than their "inner" body (extra-tympanic sound perception) during the spatialization experience. For most participants, the signal was only audible when displayed to certain positions (upper back and clavicles, i.e. those closest to the inner ear). The bi-modal aspect of the spatialization experience may therefore depend not only on a signal's position, intensity and spectral characteristics, but also on the user's conscious attention on either modality.

5.2.3. Key Questions and Limitations

When we started to design Mp2p, we tried to use an existing 3D spatializer tool: a vector-based amplitude panning (VBAP) algorithm [17]. However, our configuration did not permit this because the listening point (the head) was off-center, i.e. not based on diffusion points (speakers) placed around the origin (listener). If we use a classic calculation of spatial decay (gain = $1/\text{distance}^2$) according to airborne sound transmission, the signal disappears as soon as we move the point of diffusion away from the actuators' exact positions. Realistically, we would need to create a more specific spatialization model which takes into account the body's responses to extra-tympanic conduction and tactile stimulation of the skin, as described in section 4.1.3. The listening point (i.e. listener's position) would be the head, while the haptic sensations would be distributed: they would not refer to any central reference or origin point, since tactile perception is decentralized and perceived all over the surface of the skin (i.e. each

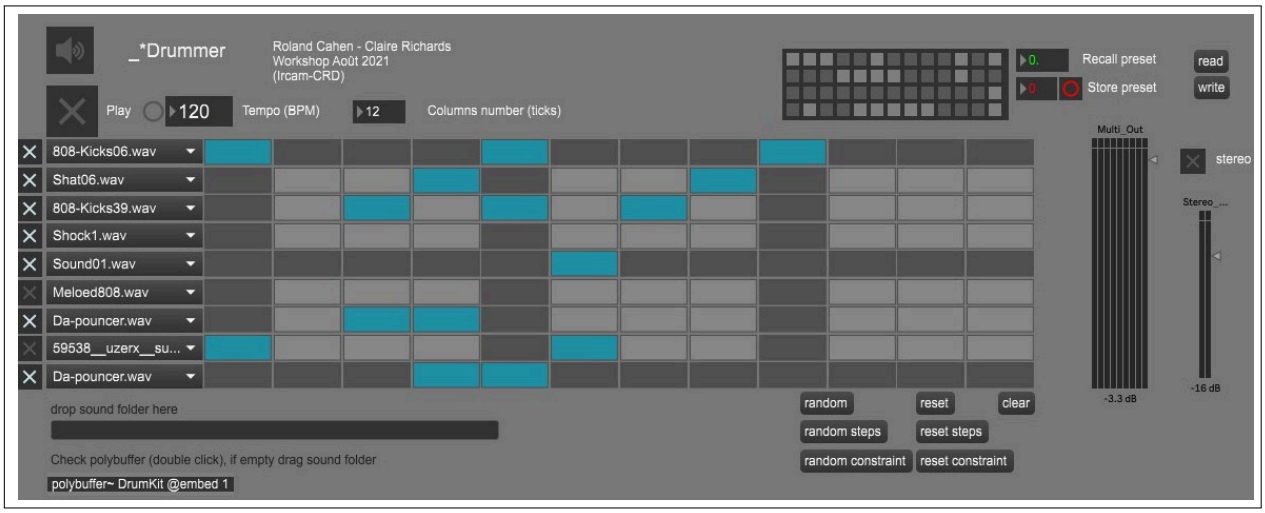


Figure 7. The Drummer interface seen in the Max environment. The sequencer tracks each correspond to one actuator position on the harness. Above the sequencer, there is a preset grid for saving specific drumming patterns.

actuator's position would have its own sensory reference).

One key limitation of Mp2p is the static nature of the configuration preset system. The signal path configurations are saved within presets and are static within a single composition. In other words, the designer cannot use multiple configuration presets in one spatialization time frame, limiting each composition to a single path that the signal can follow. One possible solution is to create a state-based programming interface which would include a configuration for each state, and a sequencer that allows the designer to edit each successive state and concatenate them with discrete or progressive transitions between states. In its current state, Mp2p does not integrate the equalizer profile function of Inner.Music, so in order to calibrate the signal intensity for the different sensitivities of each point of stimulation, designers must manually tweak and adjust the amplitude levels along the duration of the displayed signal. It is time-consuming to find the right nuances of intensity and frequency values for a given point in order to obtain the intended perceptive results. A final limitation of this tool is that the signal is mono: we cannot create "harmonies" of different sounds and movements. In section 6, we discuss some steps to address this constraint.

5.3. Drummer: Body-based Drum Machine

While we discussed musical metaphors during the workshop, we decided to create a basic nine-track step sequencer and drum machine.

5.3.1. Interface Design and Composition Features

Each track on the interface corresponds to one actuator and plays one selected sound file, imported from the user's library. The designer creates their drumming pattern by activating buttons on the sequencer, creating repetitive patterns of stimulation in which each actuator/position performs a sequence of an instrument (see fig. 7). The patterns can be saved as presets.

5.3.2. Trial Feedback

Participants described using the Drummer tool as a fun experience: the interface is simple and user-friendly, patterns can be quickly designed, and the ability to assign different instruments to the different points of stimulation is unique to this tool. Participants tended to move around while testing it, in sync with the rhythm. We heard several remarks about enhanced awareness of one's own body, and the presence of the wearable device. P.02 said, "I feel like the different instruments are playing directly on my body, or like I am the instrument!" She added that, since she heard the different sounds at the same time as she felt the stimuli on the different positions, the sound seemed to be coming directly from those points. This comment suggests that, even though the spatial audio information is lost in extra-tympanic conduction, the localized perception of tactile stimuli can contribute to impressions of auditory source positions. This comment also branched off into a discussion about sensory effects for physical rehabilitation, perhaps to help target the user's attention on a specific part of their body.

5.3.3. Key Questions and Limitations

In the Drummer interface, the opportunities to create musical patterns remain basic: the designer can select the output channels, choose sound files, alter the tempo and number of columns, but cannot change relative signal intensity according to point of stimulation. The interface functionalities could be enriched with multichannel or MIDI sequences, an equalized distribution of the signals so that their intensities can be tailored to each point of stimulation, or a more dynamic variation of each track's intensity.

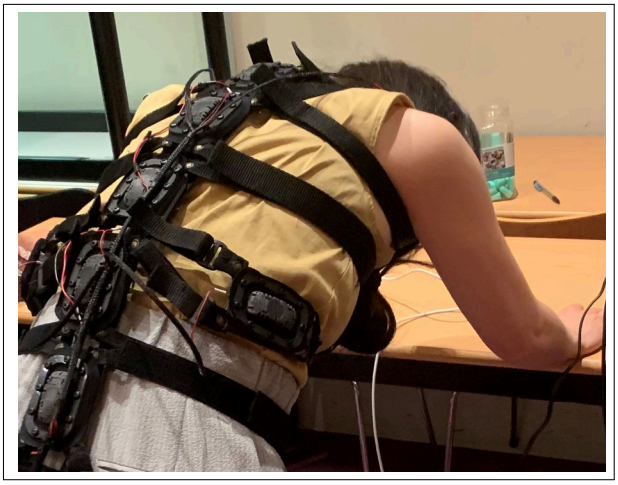


Figure 8. Photo taken during the workshop. P.02 bends forward to change her perception of the transmitted vibrations.

One could also imagine guiding the user's movements by distributing different sensations to different parts of the body. However, body movements and changes in posture interfere with audio-haptic perception. The pressure of the modules against the body changes based on the way the user stretches their joints and limbs (see fig. 8). For example, when the user bends downwards, or stretches their arms in front of them, they pull the modules closer against their spine. The added pressure on the points of stimulation, along with the difference in the curvature of the spine, facilitates good contact of the modules and thus good transmission of the vibratory signals. However, if the user raises their arms above their head, the upper back arches inwards, creating more distance between the modules and the skin and reducing the vibratory transmission. This aspect of use of the multimodal harness could be seen as a limitation but also as an advantage: though the body's movements cause variations in audio-haptic perception, those variations are part of the physical interaction between the user and the wearable, and could be a source of perceptual cues for movement-related use cases. Modifying accents, rhythm or instrument positions according to the user's movements would be interesting to explore, but this implies the integration of motion and pressure sensors which is currently out of the project's scope.

6. DISCUSSION

During the two weeks of the workshop, we were able to address each theme of audio-haptic composition (multimodal music, sensory equalization, and signal spatialization). However, we experienced developmental setbacks due to time constraints and a lack of knowledge about user perception. Evidence of these current limitations can be found in several aspects of the interface functionalities: a sensory equalizer function is present in only one of the authoring tools (Inner.Music), the spatialized audio-haptic signals are mono and limited to basic wave types, and we are restrained to one-dimensional position configurations. Although they contain some similar elements, the authoring tools are distinct from one another, and the resulting sensory experiences are limited to each interface's distinct functionalities.

These constraints are temporary: we aim to streamline and optimize our authoring tools. For example, in future iterations, features could be consolidated into one interface: the sensory equalizer in Inner.Music, the curves in Mp2p for modulation of position, frequency and amplitude, and the position-specific sound selector in Drummer. For example, in a unified interface, the designer could use the sensory equalizer to first determine a baseline audio-haptic calibration that suits their perceptual preferences. They could then use Drummer's function to select specific sound files to display at each position, and then use Mp2p's curves for spatialization pattern design to map the signal display at each position, along with more fine-tuned variations in amplitude and frequency across the duration of the composition. Inspired by existing tools for audio spatialization [18], we have already begun to optimize the mono-source design constraint of Mp2p in a separate interface (called "MC-Curv"). Instead of one single spatialized source, the designer could edit and spatialize up to four sources on the body at the same time, creating harmonies of movement across the stimulation points instead of single-path sensations.

Regarding the design of the multimodal harness, we hope to improve many aspects in the next iteration: sound transmission quality, comfort, and ease of adjustability. The second iteration of the harness involves changes for both the wearable structure and the module. The actuators will be integrated in an orthogonal position relative to the surface of the body, instead of parallel. Though they will protrude from the surface of the device, this will allow the voice-coil actuator's lengthwise-directed vibrations to penetrate the surface of the body rather than rubbing horizontally across the skin. We hypothesize the change in orientation will increase sound perception in the mid/high frequency range, bringing more clarity to certain instrumentals and vocal sounds, without impacting the quality of tactile stimulation.

7. CONCLUSIONS

Since there is no generic go-to tool for audio-haptic composition, we faced the same developmental challenges as past composers and researchers: we wanted to create and study multisensory effects, but we didn't have the tools to do so. In [2], the authors complain of the "awkwardness" of designing the vibrotactile sensory effects using Protools, a standard Digital Audio Workstation. Nine years later, in [14], the authors used Premiere Pro to create "vibetracks" for a film, by placing clips of different sine waves one-by-one at precise points along the video track. Today, researchers acknowledge that tactile composition is still "largely unexplored", that composers need to "take into

account haptic perceptual effects" and that they develop their own design tools for this purpose [19]. Resorting to these painstaking methods drastically slows down the design process, and makes it difficult for progress to happen in the domain.

Another obstacle to overcome is the general lack of guidelines about variations in both auditory and tactile sensitivity across different sites on the body, and across individuals. Established ranges of sensitivity according to each position of stimulation could help provide compositional suggestions, or presets according to each modality. To this end, in an upcoming study we will follow up on our past psychophysical research [1] and evaluate participants' responses regarding both tactile and auditory perception while using the multimodal harness.

The authors call for smoother workflow design for audio-haptic composition, accessible to the research community. We hope that the products of our workshop can foster discussions about what features and guidelines might contribute to an ideal audio-haptic composition interface.

Acknowledgments

The authors extend a special thanks to Yann Boulet (les Vertugadins) for defining the fine details of the module and the harness structure, and Fanjun Jia (Actronika) for his contributions in 3D design. We also thank Emmanuel Flety from the *Pôle Ingénierie et Prototypes* (PIP) of the IRCAM STMS Lab, les Vertugadins for their continued collaboration, and Actronika for their essential advice and technical support. Finally, we thank all of our friends and colleagues who tried using the harness. Funding for this research comes from the *Association Nationale de la Recherche et de la Technologie* (ANRT) and Actronika, in the framework of a CIFRE contract.

8. REFERENCES

[1] C. Richards et al., "Vibratory Detection Thresholds for the Spine, Clavicle and Sternum", in *Proc. IEEE World Haptics Conference*, Jul. 2021, p. 346.

[2] E. Gunther and S. O'Modhrain, "Cutaneous grooves: Composing for the sense of touch", *Journal of New Music Research*, vol. 32, no. 4, pp. 369-381, 2003.

[3] S. Merchel and M.E. Altinsoy, "Psychophysical Comparison of the Auditory and Vibrotactile Perception-Absolute Sensitivity", presented at the Intl. Workshop on Haptic and Audio Interaction Design, Lille, France, March 13-15, 2019.

[4] J.P. Bresciani et al., "Feeling what you hear: auditory signals can modulate tactile tap perception", *Experimental Brain Research*, vol. 162, no. 2, pp. 172-180, 2005.

[5] L. E. Crommett, A. Pérez-Bellido, and J.M. Yau, "Auditory adaptation improves tactile frequency perception", *Journal of Neurophysiology*, vol. 117, no. 3, pp. 1352-1362, 2017.

[6] M. Schürmann et al., "Touch activates human auditory cortex", *Neuroimage*, vol. 30, no. 4, pp. 1325-1331, 2006.

[7] H. Gillmeister and M. Eimer, "Tactile enhancement of auditory detection and perceived loudness", *Brain Research*, vol. 1160, pp. 58-68, 2007.

[8] Auer Jr, E.T. et al, "Vibrotactile activation of the auditory cortices in deaf versus hearing adults", *Neuroreport*, vol. 18, no. 7, pp. 645-648, 2007.

[9] S. Shimojo and L. Shams, "Sensory modalities are not separate modalities: plasticity and interactions", *Current Opinion in Neurobiology*, vol. 11, no. 4, pp. 505-509, 2001.

[10] S. Stenfelt and R.L. Goode, "Bone-conducted sound: physiological and clinical aspects", *Otology and Neurotology*, vol. 46, no. 12, pp. 1245-1261, 2005.

[11] C. Adelman et al., "Relation between body structure and hearing during soft tissue auditory stimulation", *BioMed Research International*, pp. 1-6, 2015.

[12] A. Baijal et al., "Composing vibrotactile music: A multi-sensory experience with the emoti-chair", in *IEEE Haptics Symposium*, March 2012, pp. 509-515.

[13] R.H. Margolis, and G.L. Saly, "Toward a standard description of hearing loss", *International Journal of Audiology*, vol. 46, no. 12, pp. 746–758, 2007.

[14] F.A. Geldard and C.E. Sherrick, "The cutaneous 'rabbit': A perceptual illusion", *Science*, vol. 178, no. 4057, pp. 178-179, 1972.

[15] R.W. Cholewiak, "The perception of tactile distance: Influences of body site, space, and time", *Perception*, vol. 28, no. 7, pp. 851-875, 1999.

[16] F.L. Wightman and D.J. Kistler, "Headphone simulation of free-field listening. I: stimulus synthesis", *The Journal of the Acoustical Society of America*, vol. 85, no. 2, pp. 858-867, 1989.

[17] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning", *Journal of the Audio Engineering Society*, vol. 45, no. 6, pp. 456-466, 1997.

[18] L. Pottier, "Holophon: Projet de spatialisation multi-sources pour une diffusion multi-hautparleurs", in *Journées d'informatique musicale*, Bordeaux, France, May 2000.

[19] L. Turchet, T. West and M. Wanderley, "Touching the audience: musical haptic wearables for augmented and participatory live music performances", *Personal and Ubiquitous Computing*, vol. 25, pp. 749–769, 2021.

Analysis and Evaluation of Visual Cues in Graphical Interpolators

Darrell Gibson
Faculty of Science & Technology,
Bournemouth University, UK
dgibson@bournemouth.ac.uk

Richard Polfreman
Faculty of Arts and Humanities,
University of Southampton, UK
r.polfreman@soton.ac.uk

ABSTRACT

Graphical interpolators provide a simple mechanism for synthesis-based sound design by offering a level of abstraction above the synthesis parameters. These systems supply users with two sensory modalities in the form of sonic output from the synthesis engine and visual feedback from the interface. A number of graphical interpolator systems have been developed over the years that provide users with different visual cues, via the graphical display. This study compares user interactions with six interpolation systems that have alternative visualizations, in order to investigate the impact that the interface's different visual cues have on the process of locating sounds within the space. We also present a dimension space analysis of the interpolators and compare this with the user studies to explore its predictive potential in evaluating designs. The outcomes from our study help to better understand design considerations for graphical interpolators and will inform future designs.

1. INTRODUCTION

A central challenge when undertaking sound design with a synthesizer is determining how to configure the synthesizer parameters to create a certain audio output, i.e. how to turn intended sonic characteristics into parameter values. All the more so as synthesizers often possess a large number of parameters with complex relationships to the sonic output.

Graphical interpolators offer a mechanism to simplify sound design by reducing control complexity through a *few-to-many* mapping between an interpolator and the parameters [1]. This is achieved by taking defined states of synthesis parameters ("presets") and associating them with locations within a 2-D graphical pane. Moving an interpolation cursor's position within the space results in new sounds being generated as the synthesizer parameters are changed by the interpolation model. This provides a mechanism to define a navigatable sound space which is constrained by the characteristics of the selected sounds, their locations in the space and the visual model used. Many different graphical interpolators have been developed, that provide users with a variety of visual cues, and in this work a number of these have been analysed and evaluated through user testing.

Copyright: © 2022 Darrell Gibson et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any médium, provided the original author and source are credited.

2. PREVIOUS WORK

Previously the authors undertook a review of a number of these interpolator systems and reimplemented them so they could be comparatively evaluated [1]. Examples of six of these are shown in Figure 1.

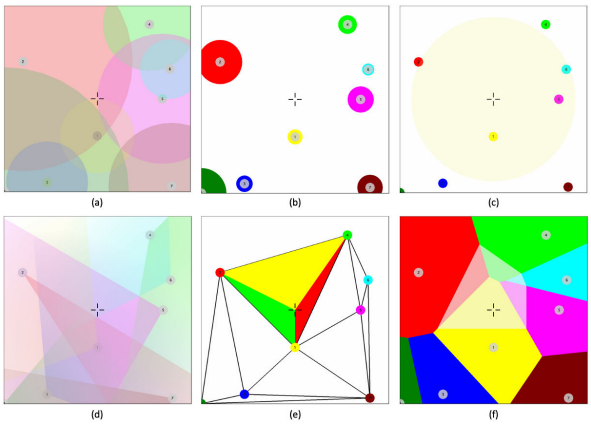


Figure 1 Visualizations for Different Graphical Interpolator Models: (a) Nodes, (b) Gravitational, (c) Radius-Based IDW, (d) Light, (e) Delaunay Triangulation and (f) Voronoi Tessellation

Figure 1a, is the *nodes* interpolator [2], where each preset is represented as a circular node and the interpolation is performed in the areas where the nodes intersect. Where there is no overlap between nodes the corresponding preset sound will be generated. The next interpolator (Figure 1b) is based on the INTERPOL control window from the SYTER system [3] which utilized a gravitational model where each preset is a *planet,* the size of which determines its *gravitational force* and so influence in the interpolation space. When positioned on a planet the gravity results in the preset sound. The third interpolator (Figure 1c) uses a radius-based Inverse Weighted Distance (IWD) model [4], where the distance to the preset locations is used for only those presets within a specified radius of the cursor. The interpolator shown in Figure 1d, uses a light model, where each preset corresponds to a lamp that emits a beam of light where the range and angle can be set. The interpolation is then performed where the light beams intersect [5]. The next interpolator (Figure 1e) uses the preset locations to form a Delaunay triangulation in the interpolation space [6]. The triangulation determines which three presets are included in the interpolation by using the vertices of the containing triangle. The relative weightings of each preset are shown as a coloured triangulation between the cursor

point and the presets. The final interpolator, shown in Figure 1f, generates a Voronoi tessellation where each polygon represents a preset [7]. The interpolation is performed between the presets that are natural neighbours of the cursor location. The relative weightings are shown by the transparent "ghost" polygon centred at the cursor.

In the previous study it was shown that different graphical interpolator models result in the generation of unique sonic pallets that impacts on the range of sounds it is possible to achieve with a particular interface and configuration [1]. A separate study was undertaken to establish whether visual cues on a graphical interface aid the navigation of the interpolation space or could the same results be obtained without any graphics (i.e. a plain interpolation space) [8]. The results from this study indicated that the number of visual cues provided by an interface does impact on the interpolators' performance and perceived usability. Tabulated analysis was then undertaken of the visual cues each interpolator model in Figure 1 provides. It was found that although the interpolators share similar goals, their visual cues can be disparate [9].

Given that the previous work indicated a link between visual cues and usability, this study examines the models in Figure 1 that provide different cues, to assess their relative performance. To do this, we first propose a mechanism for the analysis of interpolator visual cues to see if commonalities and trends can be identified between different models. User testing was then undertaken to evaluate the impact that cues may have on the usability and performance of an interface. The results should allow future designs to leverage the potential of particular visual cues.

3. INTERPOLATOR DIMENSION SPACE

Dimension space analysis has previously been defined and applied to the design of digital musical devices [10], [11], as a means of visualizing differences between systems. These have defined spaces with seven and eight dimensions respectively, but were created for somewhat different instrument design contexts, based on phenomenological versus epistemological factors. Although a graphical interpolator can be used as a musical device it does not fit with the sound design application area of interest in this work. In addition, all the interpolators have similar basic functionality so it was unlikely that it would be possible to identify significant differences using the dimension spaces previously defined. Given the area of interest here is the visual cues that each model provides, a new six-axis space is defined, shown in Figure 2. Each of the axes is marked with a representative range and is described in detail in the remainder of this section. It should be noted that although the space has been primarily created to analyse the six interpolator visualisations in this study, care has been taken in the definition of the space to ensure that it is flexible enough to analyse other graphical interpolators.

The six dimensions are derived from earlier analysis [9] and each is made up of three discrete values that describe the requirements of the visual model or how the interpolator handles a particular visual cue. Note that where there are properties that are common to all the interfaces (such as, preset locations, handles, etc.), these have not been included in order to highlight only the differences between the interfaces.

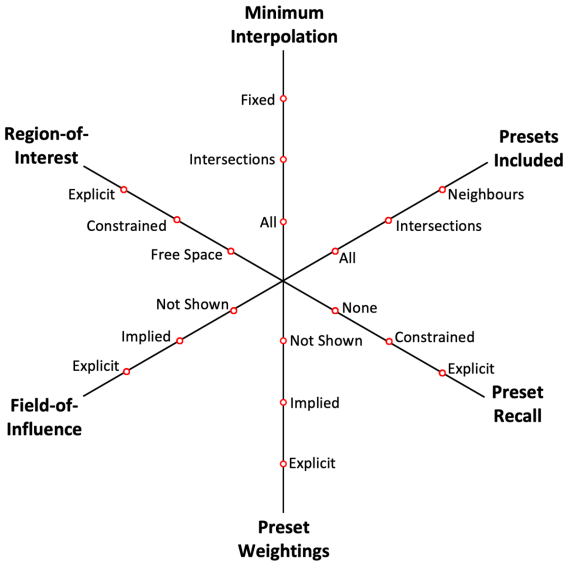


Figure 2 Six-Axis Dimension Space for Interpolator Visualisations

Where there are common discrete values on different axes, they have been positioned at the same locations and they have been arranged in order of increasing desirability, based on the findings from the testing already undertaken and the evaluation of previous systems [1, 8, 9]. In this way, the larger the dimension spaces radar plot is, the greater detail the visual cues provide on the interface. However, evaluation of the cues' actual desirability will need to be established through user testing. As far as possible, the axes have been arranged such that common values are on adjacent axes. The individual axis details follow, working from the origin outwards:

- **Minimum Interpolation** specifies the minimum requirement before the interpolation can be performed. The axis contains three discrete points: *all*, *intersection* and *fixed*. The first value, *all*, is used for those interpolators that perform interpolation between all of the presets within the space without any restriction, such as SYTER [3]. The *intersection* value is used where the interpolator needs the presets to be arranged within the space so that intersections occur, for example, as seen in the node [2] or light [5] models. The *fixed* value is used for interpolation models where a specific number of presets is required in order to perform interpolation. E.g., a triangulation model requires three presets to perform the interpolation [6].
- **Presets Included** indicates how an interpolator's visual model tells the user which of the presets within the space are included in the current interpolation output. The first value again represents the interpolation models where *all* presets within the space are included in the interpolation. The middle value, *intersections*, like the previous axis is for those systems that use some form of intersectional model, meaning the presets included in the

interpolation are those that intersect the current cursor position. The final value, *neighbours*, represents models where the interpolation is performed with presets that are neighbours of the cursor position, as in Voronoi tessellation [7].

- **Preset Recall** represents a model's ability to recall the original preset sounds unchanged. The first value, *none*, is for models where interpolation is always being performed and so it is not possible to hear the original presets. This is the case for models such as an unconstrained IDW [4]. The *constrained* value is for interpolation models where the layout of the presets may make it impossible to recall a preset. For example, with the intersecting models, such as nodes [2] the preset sound can be recalled if the interpolation point is positioned in a non-intersecting region of the preset's area. However, if the current layout does not offer a non-intersecting region, then it will not be possible to hear the original sound. The final value, *explicit*, is for models that have a specific location where the interpolation results in the recall of preset sounds. For example, with the gravitational model positioning the interpolation point on a planet's surface [3].
- **Preset Weightings** defines how the interpolator model shows the weightings of the individual presets included in the current interpolation output. *Not shown*, is for interpolators that have no visual cue to represent the individual weightings, as was the case with Interface 1 (no visualization) in the previous study [8]. The *implied* value is where the weightings are implied through the model. For example, with IDW the individual weightings are implied by the distance between the interpolation point and the included presets [4]. The *explicit* value is where the interpolator provides an obvious visual cue showing the individual weightings. For example, with INT.LIB weightings are shown visually by linking to the transparency of the presets [12].
- **Field-of-Influence** indicates an interpolator's ability to display the range of each preset within the interpolation space. The point *not shown* is used for interpolators that give no indication of a preset's range, as would be the case with interface 1 (no visualisation) or 2 (preset locations) from the previous study [8]. The next value is *implied* and is used where the range is not directly shown but is implicit by the preset's position relative to the other presets. This would be the case for interpolators such as those that use some geometric arrangement of the presets [13]. This leaves the final value of *explicit* which is used for the interpolators that show the extent of each preset, such as the light model [5].
- **Regions-of-Interest** represents how the visual model shows the areas where the interpolation is being performed. The first value, *free space* is used for interpolators that can perform the interpolation across any area that is not a preset location, as is the case for SYTER [3]. The next value, *constrained* is used where the free space is restricted in some way such as being contained as in the case for the radius-based IDW or proximity of neighbour as with Voronoi tessellation [7]. The final value *explicit* is where a region is clearly shown either by a containing shape as with triangulation interpolation [6] or as an intersectional area as with nodes [2].

3.1 Analysis of Interpolator Dimension Spaces Plots

Having defined a dimension space for graphical interpolators, this was then applied to the six reimplementations. Each was analysed against the six dimensions defined and a plot generated. For comparison these are shown in Figure 3.

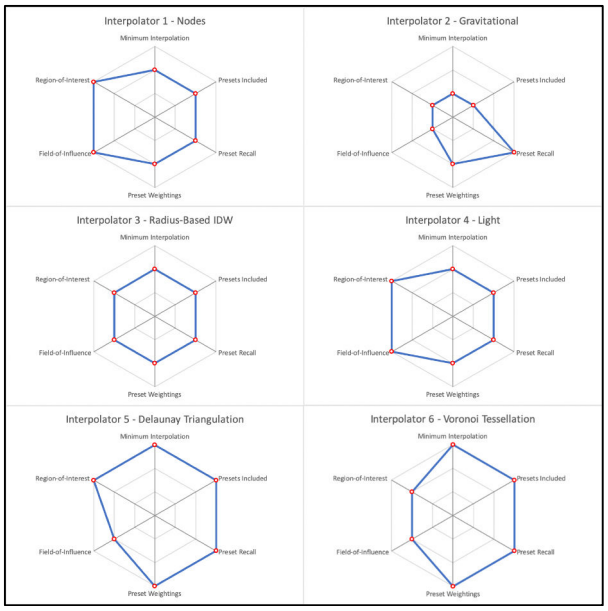


Figure 3 Dimension Space Analysis for the Reimplemented Interpolators

From this analysis it is noticeable that the gravitational model (Interpolator 2) results in a plot with the smallest area as most of the axes get the minimum score. The exceptions being *preset recall* for explicitly allowing the source presets to be recalled and *preset weighting* as this is implied by a distance function. As the values have been arranged along each axis with increasing desirability, plots that are focused on the origin could be considered less ideal than those that are wider. In this case, the gravitational model appears to be the least suitable as it does not provide the user with as many visual cues as the others. The next widest plot was for Radius-based IDW (Interpolator 3) which has the middle value on all the axes, suggesting it is preferable to the gravitational model, but not as favourable as the others. The two interpolators that both use intersecting models, nodes (Interpolator 1) and light (Interpolator 4) both produced identical plots for the defined dimension space. This is perhaps not surprising as the light model's angular component is the only significant difference between them. Both gained the highest score on two axes for explicitly showing each preset's *field-of-influence* and *region-of-interest*. The Voronoi tessellation (Interpolator 6) generated the next widest plot, having the highest value on all the axes apart from two: *field-of-influence* and *region-of-interest*, the two axes that the

intersecting models achieved the highest values on. Finally, the triangulation (Interpolator 5) produced the plot with the widest area, with the highest value on five of the six axes, but only achieving the middle value for *field-of-influence*. The fact that the last two models which featured the widest plots are the geometric duals of each other [14] may be of importance.

3.2 Evaluation of Dimension Space Results

The dimension space analysis provides a pictographic representation of the visual cues offered by each graphical interpolator. It has also allowed ranking of the interfaces against a scale of desirability and forced consideration of which characteristics might be advantageous when designing new graphical interpolator interfaces. However, it should be noted that the scale of desirability used for the plots is solely based on the result of the author's bench testing and the previous evaluation undertaken [1]. Also, with this method of dimension space analysis there is an assumption that each axis is of equal importance in the plot, which may not in fact be the case [15]. Nonetheless, the plots do offer an effective way to directly compare multiple interpolators that all have the same base functionality. To verify the outcomes from this analysis and to gather quantitative data, usability testing was undertaken, using a similar methodology to that used previously [8].

4. INTERPOLATOR EXPERIMENT

An experiment was designed to establish if there was an identifiable difference in the way that users interact with each interface. The aim was to determine if the different visual cues influence the system's performance. The same metrics used in the previous study were chosen as it was shown that the total test time, speed of cursor movement and distance moved all increased with visual cues [8]. However, in the previous study the participants were asked to locate a specific sound while in this experiment there is no "correct" location, so the user-selected locations will be used to determine the distribution of locations that produce suitable sounds. Using these metrics should provide insight into differing user interactions with each interface and through the dimension space analysis, a relationship to the visual cues the interface provides. Comparative testing was undertaken with the six reimplemented interpolators. As well as examining how users interact with each visual interface, the experiment also attempted to see if the different visual models had an impact on the sound design. To achieve this, for each interpolator the participants were given a written "brief" detailing the type of sound required, a visual reference of where the intended sound will be used and an aesthetic context for the sound. To ensure some comparability between the sound design tasks for each interface, the type of target sound was kept the same, allowing identical preset sounds to be used with each interpolator. To provide some diversity in contexts and potential sonic solutions, the type of sounds chosen were background ambiences for spacecraft in a science fiction setting. Science fiction was selected as the genre, as it requires a diverse range of sonic outputs and as it is a fictional setting, there should be less preconception of how it should sound. Spacecraft were chosen as over the years there have been many depictions of different types of spaceships: motherships, fighters, cargo freighters, shuttles, etc., which require different sonic identities, not only based on their type, but also to fit the narrative and aesthetic context. For example, the sound of the Nostromo[1] from the film Alien (1979) [16], sounds very different to the Millennium Falcon[2] from the film Star Wars: Episode IV - A New Hope (1977) [17], although they are both spacecraft depicted within a couple of years of each other. Having decided on the type of sounds, the different characteristics of each task were established. Six different goals were mapped to the six different interpolator interfaces. These were chosen to be as varied as possible so that each scenario was distinct:

Interpolator 1 - Soothing and healing sound for a medical hospital spaceship
Interpolator 2 - Manic and chaotic sound for a spaceship owned by a psychopath
Interpolator 3 - Calm and tranquil sound for a spaceship owned by a battle hero
Interpolator 4 - Threatening and scary sound for a spaceship where a killer is hunting the crew members
Interpolator 5 - Despair and despondency for the sound of a spacecraft that is stranded in deep space with no engines and dwindling life-support systems
Interpolator 6 - Sombre and gloomy sound for a dying spaceship that is being eaten by parasitic space slime

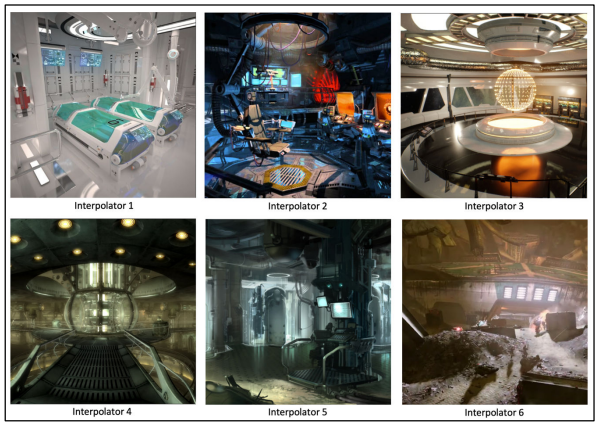


Figure 4 Visual Representations for the Tasks

To provide some focus for each scenario a visual representation (Figure 4) was supplied on an informative basis to give the participants a particular target aesthetic and make the sound design task as concrete as possible. These provided some similarity in the nature of the tasks but suggested unique sonic solutions for each of the allocated interpolators. The same ten preset sounds were set up in

[1] Recording of the Nostromo Ambient Engine Noise https://youtu.be/U4p1mZnKkhc

[2] Recording of the Millennium Falcon Ambient Engine Sound https://youtu.be/P93kbL0G0ww

each interpolator, at identical fixed locations that could not be modified by the participants. These provided a diverse range of spacecraft ambience sounds.

The participants were presented with the sound design tasks and associated interpolators in a random order. Before the experiment was started participants were given the opportunity to complete an interpolator training session, where they were introduced to interpolator functionality and operation. To avoid showing them any of the interfaces being used in the experiment, an interface was used that only showed the preset handles and had no other visual cues. The participants were given the written scenario and the corresponding visual representation, and they were then free to initiate the test once happy that they understood the sound design requirements. The interpolator/scenario allocations were the same for each participant so that comparisons could be made between them. All other aspects of the interpolation system – inputs, interpolation calculations, mappings (all parameters) and synthesis engine (Native Instruments' Massive) – remained identical between the six interfaces. As a result, sonic differences between the interpolator outputs were purely a function of the different visual models. Each test lasted a maximum of ten minutes with the participants being able to stop the test beforehand if they chose. When the participants felt that they had achieved the sound design goal, they pressed a "Target" button to register the location.

5. RESULTS

The desired number of participants for the experiment was set at thirty-six, based on a power assumption of 0.8 and the desire to observe a medium effect size (0.1758631) [18]. However, due to Covid-19 restrictions the number of participants was limited to twenty. All the participants recruited had some degree of sound design experience and all their interactions with the interfaces were captured via the recording of mouse movements. This allowed traces of the movements to be visually compared between the different interfaces. The trace gives a pictorial representation of the journey that each user made through the interpolation space. An example is shown in Figure 5.

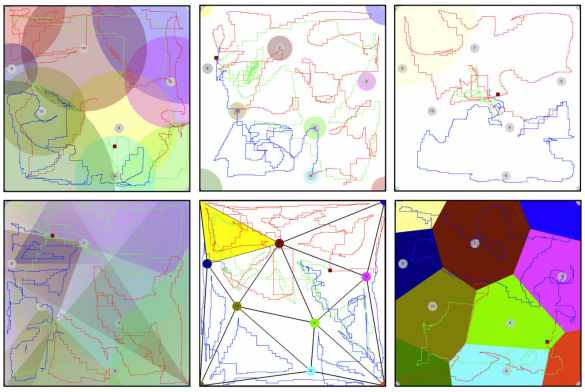


Figure 5 Mouse Traces for Participant 1 - Showing Top Row Interface 1 – 3 and Bottom Row Interface 4 – 6 and the Participant's Chosen Location (■)

To aid interpretation, the traces have been coloured so the first third of the trace is red, the next third blue and the final third green. On inspection it was found that participants appeared to follow the previously observed trend of three distinct phases of interpolation: *exploration* through making large fast moves through the space, *localisation* on region-of-interest, but occasionally checking if better options exist and *refinement* through slow small movements in the space [8]. These can also be seen by viewing a plot of cursor speed over the duration of the test. Figure 6 shows this for participant 3 in the first test they took.

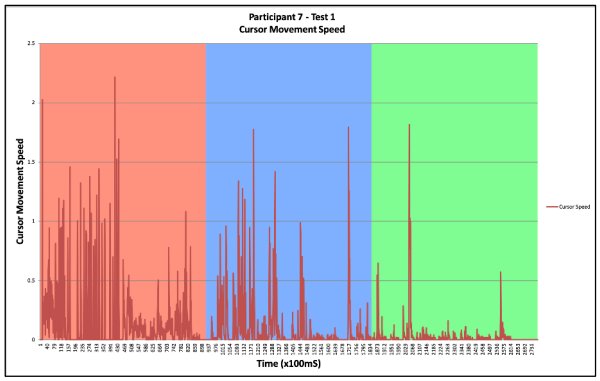


Figure 6 Mouse Speed - Sampled Every 100mS for Participant 7 with Interface 5

Although these phases do not always split evenly into thirds of the task time, many of the participants appear to follow this trend. In addition, it was noted from the traces that sometimes confirmatory moves are made in the refinement phase and slower moves are sometimes made in the exploration phase when interesting results were found. As can be seen in this plot during the first phase this participant did find an area that caused their movements to slow so that smaller distances were travelled. Also, during the final refinement phase, some faster moves were made, as normally seen when localising on regions of interest

Despite these anomalies the three phases of spatial interpolation appear to hold providing further evidence of this search behaviour being common to interpolators. It should also be noted that some participants would exhibit different *modus operandi*. For example, (Figure 7) participant 13 adopted a strategy where they undertook their navigation of the space and then afterwards, they clicked on different locations, causing the cursor to jump and audition the sound at alternative positions.

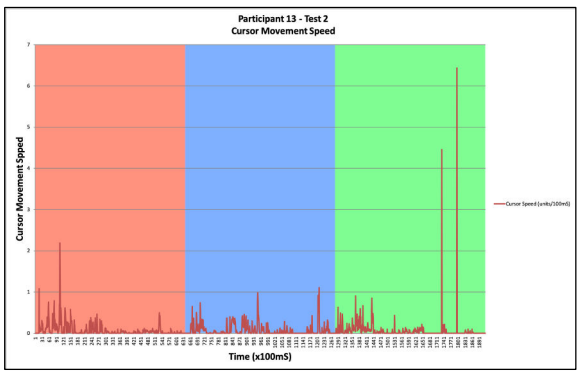


Figure 7 Mouse Speed - Sampled Every 100mS for Participant 13 with Interface 3

This example is shown for interface 3, but the participant followed the same strategy, to a greater or lesser extent, with each interface. However, aside from these cursor jumps, this participant still appeared to search the space in a similar manner. This indicates that some users have unique navigational strategies that they use across the interpolator interfaces, regardless of their graphics.

To statistically confirm the presence of the phases that have been observed here and previously [8], the mean speed of cursor movement and mean number of high-speed moves were calculated for each phase across all tests. Note that a high-speed move was defined as greater than 0.5 units/100mS. This value was chosen as it represents moving half the unit squares distance in the sample time which shows as medium spikes on the mouse speed plot (Figure 6). The results of these calculations show that both the mean cursor speed and number of high-speed moves decrease at each stage (Table 1).

Phase	Mean Cursor Speed (Standard Deviation)	Mean High-Speed Moves (Standard Deviation)
Exploration	1.373 units/sec (SD = 0.607)	40.28 mvs (SD = 38.76)
Localisation	0.981units/sec (SD = 0.433)	23.94 mvs (SD = 17.04)
Refinement	0.565 units/sec (SD = 0.422)	13.62 mvs (SD = 14.92)

Table 1 Mean Cursor Speed and Number of High-Speed Moves for the Three Phases of Interpolation

Null Hypothesis Significance Testing (NHST) was undertaken to establish if the differences between the phases were significant. It was hypothesized that during the first phase (exploration) the participants would have a higher average cursor movement speed and make more high-speed moves. These would then both reduce during the second phase (localisation) and then again during the final refinement phase (H_A: $Median_1 > Median_2 > Median_3$). Non-parametric methods were used due to the non-normal distribution of the data and the effect size was calculated using both correlation coefficient (r) [19] and probability score depth ($PSDep$) [20]. Friedman tests showed there were statistically significant differences between the phases for the average cursor speed ($\chi^2(2) = 120.5167$, $p < 0.001$) and the number of high-speed moves ($\chi^2(2) = 82.5489$, $p < 0.001$).

As a result, post-hoc pairwise Wilcoxon tests were undertaken with a Bonferroni correction which showed there are significant differences in how the users interact with the interfaces during each of the different phases of the interpolation. Using normal conventions [21], it indicates medium to large effect sizes. The results are summarised in Table 2.

Variable	Wilcoxon	Sig	r	PS_{Dep}
Speed	Z = -6.408	p < 0.001	-0.414	0.742
	Z = -8.771	p < 0.001	-0.566	0.917
	Z = -7.466	p < 0.001	-0.482	0.833
High-Speed Moves	Z = -5.256	p < 0.001	-0.339	0.667
	Z = -7.772	p < 0.001	-0.502	0.85
	Z = -6.079	p < 0.001	-0.392	0.742

Table 2 Significance Testing of Interpolation Phases for Mouse Speed and Number of High-Speed Moves

As with the previous study [8], NHST was undertaken on the mouse data to establish if there were differences between the interfaces for user interactions. Friedman tests were undertaken for the total cursor movement time, average cursor speed and the total distance the cursor moved. In all three cases the results showed no significant difference between the interfaces (Time - $\chi^2(5) = 4.886$ and $p = 0.430$; Speed - $\chi^2(5) = 6.714$ and $p = 0.243$; Distance $\chi^2(5) = 4.429$ and $p = 0.489$). Although the interfaces provide different visual cues, in these tests there is no evidence they had an impact on the participants' interactions.

The same method was used to compare the distance to the mean selected location for each interface, to determine if the distribution of selected locations could be related to the interface's visual cues. The results showed a statistically significant difference exists between the interfaces, $\chi^2(5) = 20.114$, $p < 0.001$. The post-hoc (pairwise Wilcoxon signed-rank tests with a Bonferroni correction) showed only one statistically significant difference between Interface 1 ($Median_1 = 0.319$ units, IQR = 0.400 units – 0.231 units) and Interface 6 ($Median_6 = 0.534$ units, IQR = 0.597 units – 0.399 units), Z = -2.949, $p < 0.0032$). The effect sizes ($r = -0.466$, $PS_{Dep} = 0.75$) showed a medium effect.

To further understand these results, the mean standard distance deviation was calculated to compare how much of the interpolation space was explored with each interface. This is based on the unit square size of the interfaces and are shown in Table 3.

	Mean Standard Distance Deviation (Standard Deviation)
Interface 1	0.397 units (SD = 0.051)
Interface 2	0.387 units (SD = 0.042)
Interface 3	0.412 units (SD = 0.053)
Interface 4	0.380 units (SD = 0.057)
Interface 5	0.434 units (SD = 0.067)
Interface 6	0.467 units (SD = 0.066)

Table 3 Mean Standard Distance Deviation by Interface

As can be seen, all the interfaces in this experiment resulted in higher means than those generated in the previous study [8], including the one common interface (nodes). This maybe the result of this experiment having a different goal for the participants, as in the previous study all the participants were asked to locate a single target sound with the interpolators. It is noted that the two interfaces that the dimension space analysis showed as providing the most detailed visual cues, tessellation (Interpolator 6) and triangulation (Interpolator 5), achieved the two highest scores for the standard distance deviation. Similarly, the one with the lowest standard distance deviation, light (Interpolator 4) was an interface that the dimension space analysis showed to provide less detail. This reveals that in this experiment the interface's that possess more detailed visual cues resulted in navigation of a larger area of the space.

The locations the participant selected as their chosen sounds were also plotted to see if there were trends resulting from the different interfaces. Figure 8 shows the selected sound locations for all the participants, by interface.

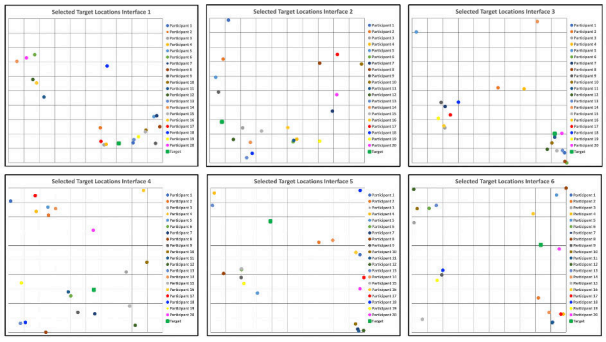


Figure 8 Participants Selected Target Locations by Interface and the Location of the Target Sound (■)

Given the subjective nature of the sound design task it is no surprise that it resulted in a wide distribution of selected locations. Nonetheless, from inspection it appears that there is some clustering of selected locations within the space. This may indicate that despite the subjective nature of sound design, there are common sonic traits that the participants identified for each scenario. From the results shown in Figure 8 the standard distance deviation was calculated with respect to the mean selected location, again based on the interfaces unit square. Hence this provides a basic measure for the distribution of selected locations (Table 4).

	Standard Distance Deviation
Interface 1	0.385 units
Interface 2	0.394 units
Interface 3	0.437 units
Interface 4	0.441 units
Interface 5	0.488 units
Interface 6	0.523 units

Table 4 Standard Distance Deviation of Selected Locations by Interface

It was noted that from these values there is again an apparent correlation with the dimension space analysis. Interface 1 had the lowest distribution of selected locations (Table 4) and, as shown in Table 3, the participants also explored less of the space. The dimension space analysis showed this interface to provide fewer visual cues in contrast to Interfaces 6 and 5 which both offer more detailed cues and resulted in users exploring more of the interpolation space and a wider distribution of selected locations.

6. DISCUSSION

From examining the mouse traces and selected locations for each interface, there is a correlation between those the dimension space analysis showed as providing more detailed visual cues, and the ones that resulted in larger distances being covered. Although this fits with what was discovered in the previous study [8], it was not possible to show a significant difference between the interfaces for time, speed and distance. Where significance was shown for the distribution of selected locations, it was only shown for one case. These results appear to indicate that although the interfaces present the users with different visual cues, these do not appear to affect the user performance when undertaking a sound design task with the interface. This might have been impacted by the number of participants recruited or from natural variation given that a confidence interval of 0.95 was used. Another potential factor might have been the fixed layout of the presets which could have been restrictive and limited exploration, especially given that experienced participants were deliberately recruited.

In this testing it has been possible to show again the presence of three interpolation phases (exploration, localisation and refinement). These had been previously observed, albeit with a limited range of interfaces [8]. However, it has now been possible to show that the phases are present for a much wider range of different interpolation interfaces. In addition, it has now been possible to show significant differences between the phases for the speed of cursor movements and the number of high-speed moves. This gives further confidence that the effect observed in the previous study is genuine and present regardless of the visual cues presented to the user. However, given that the visual cues for each interface were static and did not change during the experiment it seems that an Interactive Visualization (IV) paradigm could be of further benefit by allowing the user to change the level of visual detail on the interface during the different phases. Moreover, given that as the user gets closer to their intended location, they tend to make smaller moves and travel less distance, some form of zoom function could be advantageous to provide a finer level of control for the user, allowing more detailed sound design to be undertaken. Such an interface approach has already been shown to benefit an exploration process [22].

7. CONCLUSIONS

Given the number of participants in this study, care is required when interpreting the results. In the future, it will perhaps be possible to continue this study with more participants which may deliver a clearer view of any trends and provide greater confidence in the results. Nonetheless, the results appear to indicate that the more visual cues the graphical interpolation interface has, the wider the exploration of the space undertaken.

It should be noted that this experiment only examined the process of interpolator navigation and so all other attributes were controlled and unchangeable by the participants (preset sounds, locations, field-of-influence, etc.). As a result, the experiment results only capture a part of the overall usability of these tools and if the users were given a greater range of controls, potentially it will further affect the results. Future experiments will look to assess the impact that these additional controls provide.

It has now been shown that there is a significant difference in the user's interaction during the three phases of interpolation, regardless of the interface used. This suggests that the affect is from the process of interpolator navigation, rather than being dependent on the interface presented. It may be that this phenomenon is a result of any spatial navigation/exploration process and not unique to interpolation. In which case, the results may be applicable to many other areas where spatial searching is undertaken. Based on these findings, future work should consider the design of interfaces that provide users with visuals that facilitate the different phases of interpolation. In addition, it

is suggested that the visuals should not remain static, but should be user controllable, either allowing them to directly control the selection of different visualisations or automatically based on their interactions with the space. It is also noted that in the experiment results for the interpolation phases, the total test time was divided evenly into three. This appeared to work for many of the participants and provided a quick and easy way to identify differences in the user's interactions during the test time. However, it was seen that not all participant's interactions split evenly into thirds of the test's time and some participants appeared to move between the different phases of interpolation at different points in the process. Using a more data driven approach to define the phases, such as using the cursor speed, frequency of high-speed moves or distance moved against averages, could provide more accurate results. Given that the datasets from both experiments are available this is further analysis that will be undertaken. The results may then be used to automatically detect the phases from user interactions so the visualisation could be adapted depending on which interpolation phase the users are in.

The dimension space analysis provided a good guide for evaluating interpolator visual cues and so could be used or refined in subsequent interpolator developments to gauge the potential usability of different interface designs.

8. REFERENCES

[1] D. Gibson and R. Polfreman, "A framework for the development and evaluation of graphical interpolation for synthesizer parameter mappings," presented at the Sound & Music Computing Conference 2019, Malaga, Spain, 2019. [Online]. Available: http://eprints.bournemouth.ac.uk/32726/.

[2] "nodes. Max Reference.," ed. Cycling 74, 2016.

[3] J. F. Allouis, "The SYTER project: Sound processor design and software overview," presented at the In Proceedings of the 1982 International Computer Music Conference (ICMC), 1982, 232–240.

[4] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," presented at the Proceedings of the 1968 23rd ACM national conference, 1968.

[5] M. Spain and R. Polfreman, "Interpolator: a two-dimensional graphical interpolation system for the simultaneous control of digital signal processing parameters," *Organised Sound,* vol. 6, no. 2, pp. 147-151, 2001.

[6] K. Adiloglu, C. Drioli, P. Polotti, D. Rocchesso, and S. Delle Monache, "Physics-based spike-guided tools for sound design," presented at the Conference on Digital Audio Effects, 2010.

[7] R. Bencina, "The metasurface: applying natural neighbour interpolation to two-to-many mapping," presented at the Proceedings of the 2005 conference on New interfaces for musical expression, 2005.

[8] D. Gibson and R. Polfreman, "Analyzing journeys in sound: usability of graphical interpolators for sound design," *Personal and Ubiquitous Computing,* vol. 25, no. 4, pp. 663-676, 2021/08/01 2021, doi: 10.1007/s00779-020-01398-z.

[9] D. Gibson and R. Polfreman, "Star Interpolator - A Novel Visualization Paradigm for Graphical Interpolators," presented at the In: NIME2020: New Interfaces for Musical Expression, 21- 25 July 2020, Royal Birmingham Conservatoire, Birmingham, UK., 2020. [Online]. Available: http://eprints.bournemouth.ac.uk/34005/.

[10] D. Birnbaum, R. Fiebrink, J. Malloch, and M. M. Wanderley, "Towards a dimension space for musical devices," in *Proceedings of the 2005 conference on New interfaces for musical expression*, 2005: National University of Singapore, pp. 192-195.

[11] T. Magnusson, "An Epistemic Dimension Space for Musical Devices," in *International Conference on New Interfaces for Musical Expression (NIME)*, 2010, pp. 43-46.

[12] O. Larkin, "INT.LIB–A Graphical Preset Interpolator For Max MSP," presented at the ICMC'07: Proc. of the 2007 International Computer Music Conf, 2007.

[13] J. J. van Wijk and C. van Overveld, "Preset based interaction with high dimensional parameter spaces," *Kluwer international series in engineering and computer science,* pp. 391-406, 2003.

[14] D.-T. Lee and B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation," *International Journal of Computer & Information Sciences,* vol. 9, no. 3, pp. 219-242, 1980.

[15] M. Wójcik-Augustyniak, "How to measure and compare the value of organizations. The case study of HEIs," *Entrepreneurship and Sustainability Issues,* vol. 7, no. 3, pp. 2144-2169, 2020.

[16] G. Nis, "The Sound of Horror - Silence & Sound Contrasts in Sci-Fi Horror Movies," *Journal of Media, Cognition and Communication,* vol. 1, no. 1, 06/10 2013.

[17] G. Sergi, "Tales of the Silent Blast: Star Wars and Sound," *Journal of Popular Film and Television,* vol. 26, no. 1, pp. 12-22, 1998.

[18] D. Lakens, "Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and ANOVAs," *Frontiers in psychology,* vol. 4, p. 863, 2013.

[19] R. Rosenthal, "Parametric measures of effect size," in *The handbook of research synthesis*, vol. 621, 1994, pp. 231-244.

[20] R. J. Grissom and J. J. Kim, *Effect sizes for research: Univariate and multivariate applications*. Routledge, 2012.

[21] J. Cohen, *Statistical power analysis for the behavioral sciences 2nd edn*. Erlbaum Associates, Hillsdale, 1988.

[22] R. Tubb and S. Dixon, "The Divergent Interface: Supporting Creative Exploration of Parameter Spaces," presented at the In International Conference on New Interfaces for Musical Expression (NIME), 2014.

The Teinophon

Leo Fogadić
Aalborg University, Copenhagen
lfogad20@student.aau.dk

Dan Overholt
Aalborg University, Copenhagen
dano@create.aau.dk

ABSTRACT

This paper presents the Teinophon, a hybrid string instrument enabling alternative and unconventional playing techniques. Specifically, the Teinophon provides multiple possibilities in terms of string interaction and sound production, allowing the player to violate norms of string instrument performance. The interface features seven strings laid horizontally and parallel to one another. A string can either be plucked or it can be pulled in the upward direction, each technique resulting in a distinct articulation, and affording considerable expressive potential. The paper explains the interaction design approach, the system architecture of the interface, and parameter mapping choices. Furthermore, the sound synthesis algorithm is described, and an autobiographical evaluation is carried out. Lastly, possibilities for future improvements and features are discussed.

1. INTRODUCTION

The technology boom in recent years has stimulated the development of novel and innovative musical instruments. Due to the vast availability of microcontrollers, the development, prototyping and iteration of electronic instruments has never been easier. However, most commercial electronic instruments rely on buttons and knobs as their primary sources of interaction, wherein the familiarity with traditional instrumental techniques are fading away. In live situations, the ability for the player to expressively and artistically convey their musicality has therefore become difficult in many scenarios.

This project tries to move away from such common narratives, by allowing more natural physical interaction possibilities. Kristina Andersen states in her research into innovation processes, that if an unknown object acts or looks like a known one, it may have similar characteristics or even carry some of the original objects' essence, and elicit notions of "magical thinking" and "making strange" [1]. Andersen ask questions like: "How do we design magic?" The Teinophon resembles the look and feel of well-known instruments, but provides interactions and sounds which are impossible to achieve with traditional instruments. Its design hopes to achieve just a small bit of the whim of magic which she writes about.

Copyright: © 2022 Leo Fogadić et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. RELATED WORK

The instrument was primarily inspired by a performance project *HOMO RESTIS* (lat. "Men on Strings") by Jens Vetter and Sarah Leimcke where they used modular instrument system for live and public performance [2]. They wore full-body costumes which had strings attached to them. Opposite ends of the strings were attached to a modular system which produced sound by pulling the strings out of it. So as they were moving, different sounds were being produced and controlled by the movements.

BodyHarp, an instrument designed by Doga Cavdir, et al. [3] is another instrument that incorporates strings that can be drawn out of the instrument, in addition to the more traditional plucking techniques. The instrument is partially wearable, and its design attempts to mutualize the body and the instrument as a different way of thinking about embodied musical interfaces, in order to capture expressive dance-like body movements, as well as nuanced gestural interactions.

Other related work includes projects such as Pendaphonics, a musical installation using Gametraks devices (as were also utilized in the BodyHarp), but ceiling mounted rather than floor-based in order to function as musical pendulums. This installation created physical dynamics that can attempt to engage participants in communal musicking, be they musicians or novices. It also included a hidden affordance of plucking the pendulum strings, in order to trigger a note. This project and others utilizing similar technologies are described in Freed, et al. 2009 [4].

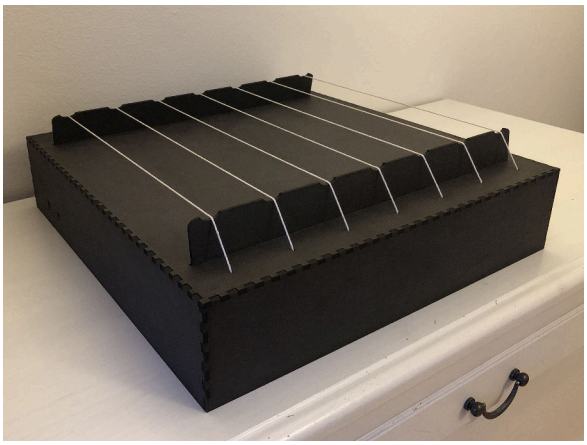


Figure 1. The Teinophon - demonstration shown at https://youtu.be/_tMA2xVPZfs

3. CONCEPT

3.1 Interaction Design Approach

One of the advantages of building a novel instrument is the ability to design the interaction with the interface. Morreale and McPherson have explained in their research in design of digital musical instruments (DMIs) that the instrument design should include "signature features" that are exclusive to that DMI, and support unique playing styles [5]. The Teinophon aims to accomplish just that, but it also aims to offer familiarity with conventional instruments (thereby providing a certain level of intuitiveness). Strings are core elements of many traditional instruments. Their length and type of material, together with the body determine the timbre of an acoustic instrument. However, strings on such instruments are by necessity stationary, with their vibrations defined by physics (excitation leads to predictable resonances). The Teinophon breaks free from the idea that strings cannot be played in other manners, by incorporating extended movements of the strings in order to produce new types of sounds and a wide range of timbres. The general concept of such disruptive design practices has been explored prior, as discussed by Lepri, et al. [6].

3.2 General Interface Design

The concept of this electronic instrument is defined by specific goals. It is meant to be used by anyone, regardless of musical experience, but also to accompany a certain level of virtuosity. Wessel, et al. [7] coined this concept as having a "low entry fee with no ceiling on virtuosity". The overall project goals are set as followed:

- **Transparency.** In order for the instrument to be accessible, it is designed while thinking about how to provide a high level of intuitiveness. This consideration means that the player should quickly be able to realize how the instrument is played.
- **Precision.** The input data and processing of the data should be precise and accurate to enable the player as much control as possible.
- **Expressiveness.** The instrument should allow many qualities of human expressivity to be portrayed (serving as a "conduit") in order for a player's virtuosity to come through, and keep their interest in using it.

4. DEVELOPMENT PROCESS

4.1 System Architecture

The instrument consist of the following electronic components: 14 potentiometers and 2 small push buttons, which are connected to a Teensy 3.6 microcontroller [1] made by Paul Stoffregen. The Teensy handles the input data and the processing of the data. The Teensy Audio Adaptor Board, which is mounted onto the microcontroller, and

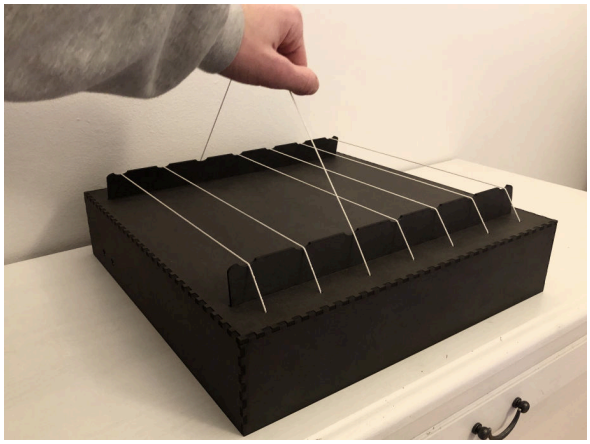


Figure 2. Interacting with the Teinophon.

TeensyAudio [2], a complete audio library for a microcontroller platform, are used to synthesise sound. 3D-printed housings [3] (see Figure 3) are holding the potentiometers in place, and the control shafts of the potentiometers are inserted into an internal part of the housing: the spool. The spool is also used to hold the ends of the strings and a metal spring in the middle. The springs are used to create tension so when the strings are pulled and released, they retract into their original position, thus turning the control shafts of the potentiometers accordingly. The spiral spring is inserted into the housing as shown in Figure 4.

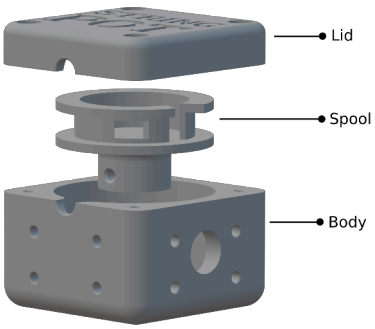


Figure 3. 3D model of the housing.

4.2 Prototype Instrument Design

All of the parts of the instrument are put into their predetermined positions in the simple rectangular box. The box was laser cut after the design was made using Autodesk Fusion360, and Adobe Illustator. The material used is a high-density fiberboard in black color with 3 mm thickness. Positioning of all the elements is shown in Figure 5. The components are held in place using velcro tape.

[1] https://www.pjrc.com/store/teensy36.html

[2] https://www.pjrc.com/teensy/td_libs_Audio.html

[3] https://www.thingiverse.com/thing:4329275

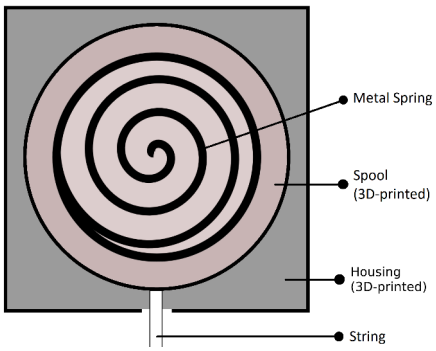


Figure 4. Representation of the inner part of the 3D-printed housing where the spool with the spring is visible.

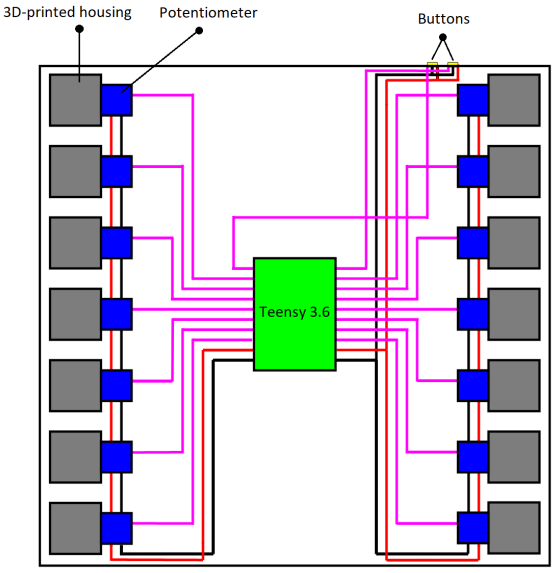


Figure 5. Simplified representation of the assembled Teinophon, with connections to the spools/potentiometers (each string connects to two of them) .

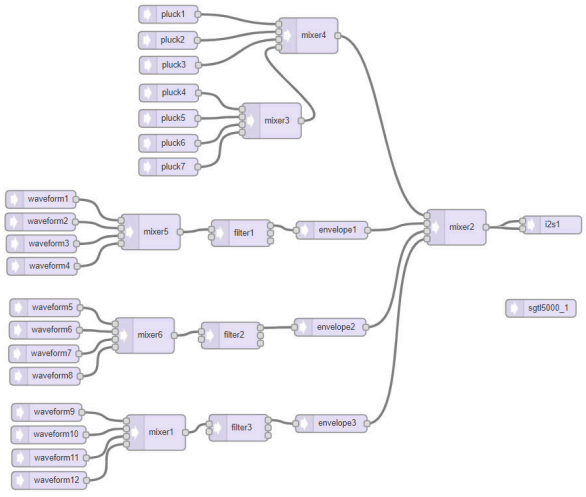


Figure 6. Teinophon's audio synthesis algorithm, as designed in `https://www.pjrc.com/teensy/gui/`.

4.3 Audio Algorithm

Using the Audio System Design Tool (see Figure 6), the initial phase of the algorithm was easy to implement. The first type of sound, triggered by plucking, is implemented using the Karplus-Strong synthesis method. The second type of sound is produced by combining 4 sawtooth waveforms, the first with the fundamental frequency, while the rest are integer multiples of this fundamental thereby providing a richer sound. There is also a low-pass filter and an amplitude envelope applied. All of the strings can be plucked at the same time, but the sustained sound can be produced only by 3 strings simultaneously in this prototype (due to limited memory of the microcontroller). However, in practice, this has proven not to be an issue, since it is rather impractical to pull more than 3 strings simultaneously.

4.4 Parameter Mapping

As each string is moving the control shafts of two potentiometers opposite to one another, the inputs from those two potentiometers are used simultaneously to control the sound. Each string is tuned to a specific frequency, based on a chosen key of a modern Western scale. On the side of the instrument there are two buttons used for changing the key. One button switches between major and minor scales, while the other button is changes the key. The range of key changes is two octaves. By pulling the string slightly and releasing it quickly (plucking), a first, short sound is produced at a frequency that corresponds to that string. The amplitude of the sound is dependant on the position of the string when it was released, in order to imitate actual plucking. When the string is raised more that approximately 5 cm from its original position, a sustained sound is produced which has the frequency of the corresponding string. Pulling the string further controls the cutoff frequency of a low-pass filter for that string. When a string is raised, higher frequencies are passed through. Lowering the string will then eliminate higher frequencies proportionally to the height at which the string is held.

5. EVALUATION

Evaluating the experience of playing the Teinophon was approached according to autobiographical design methods (ABD). ABD design research draws on genuine reflection on experience by those creating or building a system [8]. The method ensures an open-ended approach at the early stages of the designing and prototyping a new system, and captures personal perspectives on new interaction methods, through the lens of bespoke instrument designs with which the authors play, explore, and continue iteratively developing.

Reflections on playability and performance of the instrument were based after hardware assembly and over a one month period, during which the audio algorithm was also developed, tested, and refined.

During the evaluation period, there was no noticeable latency when interacting with the instrument. Although the feel of each string is not identical to one another due to a

slight difference in string tensions, the playability of the instrument is satisfactory. Interacting with multiple strings simultaneously also meets expectations; multiple strings can be played simultaneously with both techniques. However, both techniques cannot be performed on one string simultaneously. For example, the string cannot be pulled out and then plucked. Only after the string returns to its resting position, can it be plucked. The material of the box encapsulating the instrument is a lightweight high-density fiberboard. Because of that the box can move slightly on the surface when interacting with the instrument, which can be distracting.

That said, playing the instrument feels intuitive and ergonomically satisfying. Even though the responsiveness of plucking the strings is not on the level of some acoustic string instruments (the strings do not actually vibrate), after learning and acquiring a certain playing technique, strings can be plucked and give the expected sound. The strings need to be displaced to a certain level from their resting position and then quickly released to give the desired sound. This could be improved by further refinement of the pluck detection algorithm. The alternative playing technique, resulting in a sustained sound, always works as expected. After a certain level of string displacement, the resulting sound is output and the low-pass filter control is very responsive. The buttons on the side of the instrument are used to change the key and switch between the major and minor Western scales. Locking the strings to a Western scale offers familiarity, but it is also a constraint. Having a larger selection of scales, and the option to tune the strings to any frequency, would allow more tonal freedom.

6. CONCLUSION & DISCUSSION

This prototype does meet its predefined goals after the iterative development process described. The similarity with traditional string instruments makes the plucking interaction intuitive, and the novel pulling interaction is interesting and easy to perform. Therefore, playing it does not require any complicated skills. The plucking interaction, however, could be improved for an even more natural playing style. This could be improved through better data processing and algorithm refinement. The current state of the instrument allows the player to express their musicality by playing sounds and melodies with different timbres and amplitudes, thus enabling it to be used as a creative tool for sparking new musical ideas, but the audio algorithm could be improved to reflect more subtle details when playing. For example, controlling the decay time when plucking the string, depending on how the string was plucked. Furthermore, a better choice of materials for the body would make the instrument more sturdy and robust.

A useful feature that could be added to the Teinophon would be allowing the player to precisely tune every string. The player could, with a press of a dedicated button, move a string from one side to the other (allowing one end to retract more than the other), and thereby set the frequency of the string. Another expressive feature would be pitch-bending, where the player would move the strings, in the same way as tuning them, to control the continuous pitch of a string while playing. Furthermore, an abundance of different sounds could be used, since the strings can be pulled upwards up to around 40 cm from their original positions. Different levels could morph into new sounds, as the player pulls the string higher, and the sounds would change from one to the other. Finally, more sensors could be added to the system (such as an ultrasonic sensor on the side of the instrument), which would allow the player to use the motion of their hands to control various audio effects or other sound manipulations.

The Teinophon is an innovative electronic string instrument supporting players' musical expressivty. Although the first prototype presented here still has a plenty of room for improvement, it has great potential for continued development, and can already be used in a variety of musical scenarios. It has a wide target audience, since musical experience is not necessarily required to play it, and our initial autobiographical evaluation found both areas for improvement as well as areas in which it excels. The prototype Teinophon allows musicians interested in alternative forms of interaction to convey their creative ideas through the string-based instrument, by means of interesting new techniques for performance. The participation to the conference was supported by the European Art Science and Technology Network (EASTN-DC).

7. REFERENCES

[1] K. Andersen, "Making magic machines," in *10th European Academy of Design Conference - Crafting the Future, Gothenburg, Sweden, 17 – 19 April 2013: Crafting the Future*.

[2] J. Vetter and S. Leimcke, "Homo restis - constructive control through modular string topologies," *NIME '17*, 2017.

[3] D. Cavdir, R. Michon, and G. Wang, "The bodyharp: Designing the intersection between the instrument and the body," 07 2018.

[4] A. Freed, D. McCutchen, A. Schmeder, A.-M. Hansen, D. Overholt, W. Burleson, C. Jensen, and A. Mesker, "Musical applications and design techniques for the gametrak tethered spatial position controller," 01 2009.

[5] F. Morreale and A. Mcpherson, "Design for longevity: Ongoing use of instruments from nime 2010-14," 05 2017.

[6] G. Lepri, A. McPherson, and J. Bowers, *Useless, Not Worthless: Absurd Making as Critical Practice*. New York, NY, USA: Association for Computing Machinery, 2020, p. 1887–1899.

[7] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," 10 2020.

[8] C. Neustaedter and P. Sengers, "Autobiographical design in hci research: Designing and learning through use-it-yourself," in *Proceedings of the Designing Interactive Systems Conference*, New York, NY, USA, 2012, p. 514–523.

A NEW SENSOR TECHNOLOGY FOR THE SONIFICATION OF PROXIMITY AND TOUCH IN CLOSED-LOOP AUDITORY INTERACTION

Pascal Staudt (pascal.staudt@hu-berlin.de), **Anton Kogge**, **Marcello Lussana**, **Marta Rizzonelli**, **Benjamin Stahl**, and **Jin Hyun Kim**

Humboldt University, Berlin, Germany

ABSTRACT

The Italian word sentire [sen'ti:re] means both to hear and to feel. What if one could "hear" closeness or distance to another person? The paper presents a new sensor technology for sonifying proximity and touch which we have developed for Sentire, a digital system that mediates between bodily movements and musical sounds. Using this technology, inter-body proximity and touch can be sensed wirelessly and mapped to an algorithmic sound environment in real time, allowing for a closed-loop auditory interaction between two or more people in a physical environment. We describe its iterative and incremental development within the interdisciplinary research project "Social interaction through sound feedback–Sentire," which combines (artistic) human-computer interaction, sonic interaction design and real-world research. The prototype and its underlying sensor technology—based on electro-quasistatic coupling—have been successfully used in couple therapy, serving as both a framework for usability testing and to investigate the relationship between sound, sociality and therapy.

1. INTRODUCTION

Since the implementation of protective measures to counter the global COVID-19 pandemic, interpersonal distance has gained new implications. So-called social distancing caused a shift in perceptions of personal space. Proxemics, a field that studies the human perception of personal space, dates back to 1966, when anthropologist Edward Hall classified four culturally dependent zones of interpersonal distance: intimate (less than 0.5m), personal (0.5m–1.2m), social (1.2m–3.6m) and public (3.5m–7.6m). According to Hall [1], these zones strongly influence how people interact with each other. Greenberg et al. [2] point out the relevance of Hall's studies in Ubiquitous Computing. Although they refer more generally to inter-entity distance rather than inter-body distance, they underline the necessity, and possibilities, of measuring proximity for interaction design.

Established systems for measuring physical proximity and using it in the digital domain typically rely on infrared or ultrasonic sensors. These systems measure the distance at discrete points and require a clear field of vision. They are thus disadvantageous for dyadic interaction. The sonification of inter-body proximity includes different research fields ranging from (sonic) interaction design, including sound design [3–8], and embodied interaction [9, 10] to human-computer interaction design [11, 12]. Although these fields all contribute to the interdisciplinary character of our research, this paper focuses on the aspects of the underlying sensor technology developed within our project, called "Social interaction through sound feedback – Sentire" to systematically investigate its applicability in the context of therapy.

First, we present related work and the theoretical background of proximity sensing based on inter-body coupling. Next, we describe the soft- and hardware we developed over an iterative and incremental process. Finally, we present the results of prototype testing in couple therapy [1] sessions.

Copyright: © 2022 Pascal Staudt et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. RELATED WORK AND THEORETICAL BACKGROUND

2.1 Sensing Human Proximity and Touch

At the time of writing, we are unaware of any other sensor system used to sonify proximity and touch between humans as the subject of academic research. However, there have been interfaces developed that explore the relationship between touch and sound feedback or that use proximity sensors in interactive scenarios. *Skintimacy* [13] uses skin conductivity to sonify touch events between two or more humans; Müller et al. investigate its influence on intimate behaviour in the context of performance and musical interaction. A similar system, called *Closer*, was developed for examining embodied interaction in health care, though no results were published [14]. *Freqtric Drums* is a musical interface that can quantitatively detect skin contact by measuring skin resistance which is related to touch pressure and surface area [15] [2]. Hachisu and Suzuki developed smart bracelets to connect interpersonal touch with vibrotactile feedback [16]. The *Expressive Wearable* [17] uses, among other measurements, proximity sensors to express the wearer's attitude; although designed for social interaction, it uses visual feedback and does not utilise the body as an interface. *TONG* [18] uses optical proximity sensors and visual feedback to remind people of interpersonal distance.

[1] Sometimes called couple's therapy or couple's counseling.

[2] TOUCHME by playtronica (https://playtronica.com/) is a commercial product for sonifying touch, but is not the subject of academic research.

Although optical sensors like Microsoft's *Kinect* or Intel's *RealSense* and standard proximity sensors (e.g. ultrasonic or infrared-based) can detect proximity, they rely on a clear field of vision and measurements at discrete points. Therefore, they are less convenient for social interaction, which requires freedom of movement independent of sensor positioning. Bian et al. however has developed a magnetic field proximity sensor for social distancing, which measures sensor-to-sensor distance rather than inter-body proximity [19].

Sentire overcomes the limitations and problems introduced by measuring proximity at discrete points, since the body itself becomes part of a sensor system that works regardless of the location of the measurement and orientation of the body in the space. The wired version of Sentire is based on a sensor system developed by Jonathan Reus [3] and the *Metabody* project [20]. This prototype was further developed for the participatory performance series Sentire, which had been presented at numerous international festivals and events [21], before it finally evolved into an academic research project.

2.2 Inter-body coupling

Sentire's wireless proximity sensors rely on electro-quasistatic coupling between two human bodies. The phenomenon of the human body to electrically couple—which occurs because of its conductivity—is researched across disciplines. The focus is usually on *intra-body communication* [4] (IBC)—signals transmitted inside the same human body [22, 23]—or on the coupling between humans and objects [24–26]. Other related research focuses on the characteristics of the human body as an antenna [27, 28], the transmission of digitally encoded information in *body-to-body sensor networks* (BBNs) [29] and the preconditions and properties of the body as a communication channel [30].

To our best knowledge, at the time of writing, we are the first to exploit and research the effect of inter-body coupling for proximity sensing in the academic context. Grosse-Puppendahl et al. investigate capacitive near-field communication for ubiquitous interaction between persons and everyday objects [25]. A similar approach uses wrist-worn capacitive sensors to detect human touch, proximity and activity but is limited to movement-based actions between humans and objects [26].

Nath et al. [31] were the first, to their knowledge, to develop a theory of human inter-body coupling relying on *electro-quasistatic human body communication* (EQS-HBC). Their model serves as a theoretical framework to understand the mechanism behind the wireless proximity sensor we developed. Figure 1 illustrates the electro-quasistatic (capacitive) coupling between two human bodies as described in [31]. Although Nath et al. investigate inter-body coupling for analysis of security and interference properties in EQS-HBC, as opposed to describing its potential for proximity sensing, their theory and findings apply to the use of Sentire. Nath et al. provide a simplified circuit model

[3] https://jonathanreus.com

[4] Sometimes referred to as human body communication (HBC). For a review of research on HBC, refer to Zhao et al. [22].

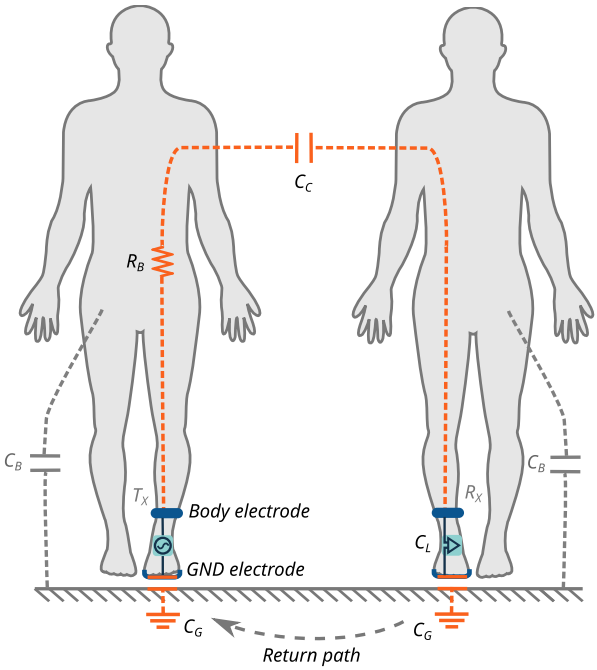


Figure 1. Capacitive inter-body coupling. Adapted from [31] which uses floating electrodes for EQS HBC; for Sentire's proximity sensing we use electrodes attached to the shoes.

from which they derive an expression for the channel loss with a capacitive load at the receiver (Rx) and frequencies below 1MHz:

$$\frac{V_o}{V_i} \approx \frac{C_{G,Tx}}{C_B} \cdot \frac{C_C}{C_B} \cdot \frac{C_{G,Rx}}{C_L} \tag{1}$$

Where $C_{G,Tx}$ and $C_{G,Rx}$ denote the capacitance between ground electrodes and ground at the transmitter (Tx), respectively, the receiver (Rx); C_B, is the capacitance of the bodies coupling with the environment, C_C is the capacitance between the two bodies and C_L is the capacitive load at the receiver. The coupling capacitance C_C depends on the distance of the two human bodies and therefore is proportional to the channel loss:

$$\frac{V_o}{V_i} \sim C_C \tag{2}$$

Consequently, the measured voltage V_o at the receiver can be used to derive an inter-body proximity signal as described in the following section.

3. DEVELOPMENT

3.1 Technical Implementation

The foundation and starting point for developing the wireless proximity detection was an existing version of the Sentire sensor system, which uses two audio cables and conductive bracelets in conjunction with a consumer-grade audio interface and custom-developed software written in SuperCollider. Although in both the wired and wireless versions the physical phenomenon of capacitive coupling

enables proximity sensing—i.e. the detection of distance and touch between the two human bodies—only substantial new hard- and software development rendered a wireless prototype possible. Before we describe the technical details and the primary efforts in the iterative process, we here provide an overview of the technical implementation.

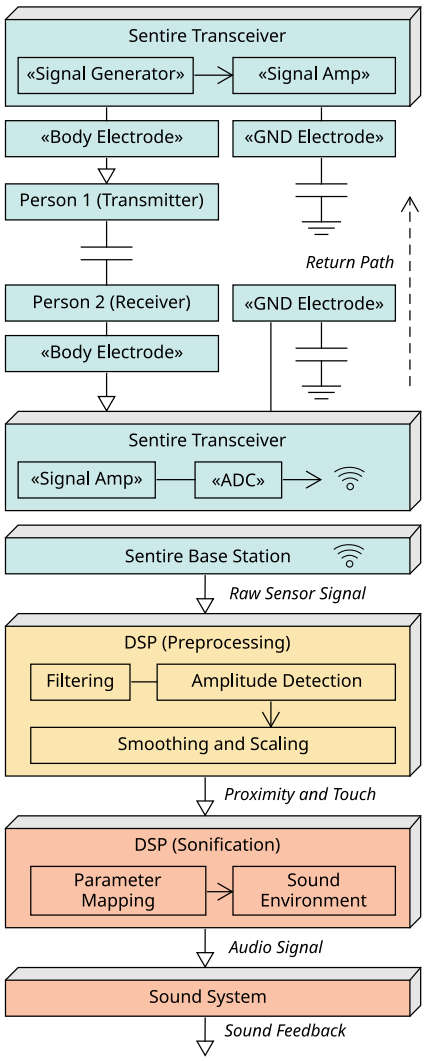


Figure 2. Schematic diagram of Sentire, including–from top to bottom–the wireless proximity hardware (blue), the sensor data processing (yellow) and sound generation (red).

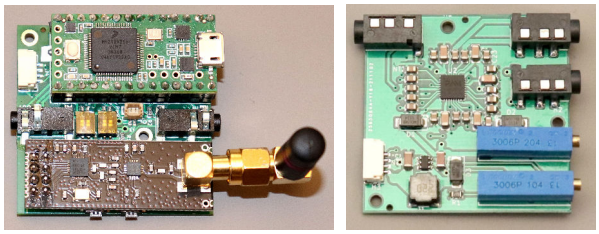
Figure 2 illustrates the various components and indicates the signal path of our closed-loop interactive interface. The interacting persons ("Person 1" and "Person 2") wear custom-developed electronic devices ("Sentire Transceiver") connected through body electrodes on their skin. Furthermore, they wear so-called "Ground Electrodes" attached to the soles of their shoes. We use a signal generator and an amplifier to feed an electrical signal through the body electrode into Person 1 (called the transmitter for convenience). As described in section 2.2, capacitive coupling of the two bodies causes a signal transmission to the receiver (Person 2), whereby the return path of the signal goes through the ground electrodes, which capacitively couple with the environmental ground. The strength of the received and amplified signal depends on the distance between the receiver and the transmitter (simplified, one can think of two capacitor plates). We utilise this "proximity effect" to generate a control signal, which changes linearly with the distance of the two bodies. The transceiver at the receiver amplifies the transmitted sensor signal and sends it to a base station via digital radio. To obtain and linearise the control signal in the digital domain, we use our software written in *SuperCollider*. These digital signal processing (DSP) stages that convert the "raw" sensor signal into a proximity and touch signal are part of the proximity sensor (i.e. the gestural input) and are described in more detail in section 4. In contrast, despite using the same software framework, the sonification stage is part of the sound generator following the classical definition of digital musical instruments [32]. Mapping the proximity signal to selected parameters of an algorithmic sound synthesis process and providing auditory feedback through a sound system finally enables closed-loop auditory interaction.

3.2 Hardware Development

The iterative hardware development of the Sentire prototypes ("transceiver") was foremost influenced by the need to enable wireless proximity and touch sensing comparable in range and reactivity to the wired Sentire system. A second premise was the usability of the prototype in the context of couple therapy [33]. Besides implementing the technical requirements, this necessity also meant designing wearable and failsafe prototypes for our real-world studies. The limited technical and human resources required us to focus on a straightforward and pragmatic design. The individual components were kept as modular as possible to enable parallel development and save resources. After testing different platforms and frameworks in the first iteration, we decided to use a *Teensy 3.2* microcontroller development board, together with an *SGTL5000* Audio Codec. The Teensy board offers flexibility and fulfils the requirements for most of the wireless sensor components (signal generator, signal receiver, radio transmitter and base station). Another advantage of using the same microcontroller platform for different purposes is to minimise the development effort (e.g. PCB design and part sourcing only needs to be done once instead of three times). The prototypes are powered by a *250mAh Li-Po* battery.

3.2.1 Transceiver Board

Figure 3a shows the developed Sentire transceiver board assembled with a Teensy 3.2, an SGTL5000 Audio Codec and an *nRF24L01+* radio module for the wireless transmission between the receiver and the base station. The transceiver board is used in different stages of the wireless sensor system (see figure 2): (1) At the transmitter, generating the transmitter signal; (2) at the receiver, amplifying the received signal and sending it to the base station via radio; (3) at the base station, receiving the "raw" sensor signal.



(a) (b)

Figure 3. Sentire transceiver (a) and signal amplifier (b) circuit boards.

3.2.2 Signal Amplifier

One way to improve the maximum distance of the proximity detection is to increase the power of the transmission signal. The signal amplifier's operating voltage and gain factor determine the maximum peak-to-peak voltage at its output and, therefore, the transmission signal's power. The regulated output of the Teensy Board delivers 3.3V (at a maximum of 250mA). Using a DC-DC step up converter based on the *MT3608* chip (up to 30V) and in conjunction with a *PAM8006ATR* class-d amplifier (8 to 18V supply) allowed for an increase of 14.7dB in signal strength compared to using the regulated output of the microcontroller (see figure 3b). Another attempt to increase the transmission power was made by running at 3.3V operating voltage and using a transformer at the output of the amplifier. Although we could increase the output voltage, the transformer itself caused an unwanted side effect: direct coupling between transformer and receiver, and, therefore, disabled proximity detection based on capacitive inter-body coupling.

3.2.3 Electrodes

One of the biggest challenges in developing the prototype as a wearable was designing and manufacturing suitable ground electrodes. We assume that the ground electrodes form a parallel-plate capacitor with the ground and therefore the following formula applies:

$$C = \epsilon_0 \epsilon_r \cdot \frac{A}{d} \tag{3}$$

Where A is the surface of the electrode and d corresponds to the distance between ground electrode and physical ground. From (3) follows:

$$C_G \sim \frac{A}{d} \tag{4}$$

And with equation (1) from section 2.2 furthermore:

$$\frac{V_o}{V_i} \sim \left(\frac{A}{d}\right)^2 \tag{5}$$

This means the transmission ratio $\frac{V_o}{V_i}$ is proportional to the square of ground electrode area A and inversely proportional to the square of distance d between ground electrode and ground. Consequently, one premise was to maximise A and minimise d. On the basis of rapid prototyping and qualitative user tests, several requirements were set:

1. Electrodes are attached to the shoes (this requirement arose from the context of couple therapy, where some clients find it unpleasant to take off their shoes)
2. Minimum distance d, made possible by attaching the ground electrode to the shoe soles and minimising the thickness of the overlay material
3. Thin yet sturdy and non-slippery overlay material
4. Tight grip on the shoe
5. Flexible fixture to fit different shoe forms and sizes
6. Feasibility with the given resources of the project

Several iterations of material research and prototype testing resulted in an electrode based on silicone injection moulding and flexible conductive fabric. In an attempt to design a single-component foot electrode, we used *single-walled carbon nanotubes* (SWNCTs) as an additive (1 percent) to the *liquid silicone rubber* (LSR). The blended SWNCTs make the rubber itself conductive yet maintain its flexibility. Although this iteration served as a proof of concept, we rejected it because we could not estimate the material's biocompatibility and health risks. [5] Therefore, we used the previous iteration of silicone electrodes. The moulds for both electrode iterations were made from *polylactic acid* (PLA) using a *fused deposition modelling* (FDM) 3D-Printer.

While describing all the technical details of the production process goes beyond the scope of this paper, it's worth noting here that the body electrodes had relatively few requirements compared to the ground electrodes described above, and were thus relatively easy to implement. After using standard *electromyography* (EMG) electrodes in the first iterations, we designed body electrodes based on flexible conductive fabric and velcro (see figure 4).

Figure 4. Sentire wireless prototype.

4. SOFTWARE DEVELOPMENT

The custom Sentire software was written in SuperCollider and handles an essential part of the wireless proximity de-

[5] Studies indicate high bioaccumulation [34] and possibly asbestos-like toxicity [35].

tection.Though developed in parallel and as part of the software, the specific parameter mapping and implementation of our algorithmic sound environments are not described in detail in this paper.

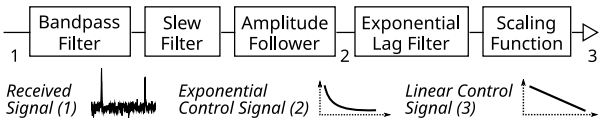


Figure 5. Digital signal processing (DSP) chain for obtaining the proximity and touch signals.

Figure 5 shows the DSP chain from the received "raw" sensor signal to the obtained proximity (and touch) signal, mapped to an algorithmic sound environment. To extract the transmitted signal and remove noise, the chain filters the sensor signal twice (bandpass and slew filter). Then, an envelope follower detects the amplitude and gains an exponential control signal dependent on proximity. Two further DSP blocks smooth ("Exponential Lag Filter"), linearise and scale ("Scaling Function") the signal and yield a linear, normalised control signal.

One major disadvantage of relying on ground electrodes is the strong dependency of the signal strength on distance d (see formula (5)). Thus, raising the foot will cause the signal amplitude to drop and, as a result, falsify the proximity signal. Our attempt to compensate for this effect is to use two receivers and two transmitters, one attached to each foot. We applied a bandpass filter to the signals sensed at the receivers using the individual transmitter signal frequency to return four sensor signals. Assuming that at least one foot is always on the ground, we select the strongest sensor signal for proximity detection. Yet, duplicating the sensor hardware introduced another problem: cross-coupling between transmitters and receivers. Table 1 shows the resulting influence of raising a single foot on the four sensor signal channels.

	Signals at receiver one (R_1)		*Signals at receiver two (R_2)*	
	S_1	S_2	S_3	S_4
$R_1 \uparrow$	down+	down+	up	up
$R_2 \uparrow$	up	up	down+	down+
$T_1 \uparrow$	down+	up	down+	up
$T_2 \uparrow$	up	down+	up	down+

Table 1. Influence of raising individual feet on the four sensor signal channels (S_1–S_4)

Raising one foot at the receiver side (e.g. $R_1 \uparrow$) strongly dampens ("down+") the sensed signals (S_1 and S_2) on this side , while it raises ("up") the sensed signals at the second receiver (other foot, S_3 and S_4). The other receiver changes the signals analogously. In contrast, raising one foot at the transmitter side (e.g. $T_1 \uparrow$) strongly dampens signal one (S_1) at receiver one (R_1) and signal three (S_3) at receiver two (R_2), while lifting signal two (S_2) at receiver one (R_1) and signal four (S_4) at receiver two (R_2). Because every combination of changes is mutually exclusive, we can algorithmically detect distinct foot movements (up, down) and distinguish them from proximity changes. Using thresholds for fast changes makes the process more robust. The algorithm compensates for the influence of footsteps to a great extent by multiplying the individual signals with correction factors that have been determined empirically.

5. FOSTERING SOCIAL INTERACTION THROUGH SOUND FEEDBACK

The sonification of proximity and touch as used for Sentire is influenced mainly by the experience with the system in the artistic context of participatory performances. Within the interdisciplinary Project "Social Interaction through Sound Feedback," our sonic interaction design had a strong focus on clinical and non-clinical therapeutic use cases.

In line with [36], we call the set of parameters that defines the sonic quality a "sound environment" (SE) [37]. A SE results from predefined artistic decisions concerning sound design and algorithmic composition combined with a specific parameter mapping. Sentire's SEs shape the interactive experience. We therefore developed the SEs by focusing on kinaesthetic perception—i.e., perception of one's own movements—and by intensifying the interaction in the intimate and personal space. According to Joseph Rovan et al.'s definition of parameter mappings [38], we use a "divergent" mapping that links a one-dimensional gestural parameter (proximity) to multiple musical parameters simultaneously [37].

Sentire provides several different SEs with distinct sound qualities ranging from atmospheric and tonal sounds to percussive and rhythmic, each aiming to sonically intensify the acts of approach and touch. We developed an SE for our couple therapy study based on the physical modelling of singing bowls. The interaction usually starts with silence and at a distance of about $3m$. As the participants move closer, the sound's amplitude rises; furthermore, the fundamental frequencies' distribution and harmonics shift from slightly inharmonic ratios (e.g. 3:4, or perfect fourth) to more harmonic intervals (e.g 2:3, or perfect fifth). Touch events stop the continuous sound generation and trigger one of three chords (Cm9, Fm11 and E♭maj9) together with pseudo-random sequences derived from the tone material of these three chords (without C and D, spanning two octaves). Consequently, the first note played determines the musical mode. We source the sounds for touch sonification from Fender Rhodes (chords) and Celli (melody) synthesizers implemented in SuperCollider. Sustained touches keep the melody playing, while the duration of touch alters the tempo of the played notes.

The couple therapy sessions served both as a framework for usability testing and to investigate the relationship between sound, sociality and therapy, combining Structured Observation and Microphenomenological Interviews. It is worth noting that our wireless technology for the sonification of proximity and touch can be used in a broad range of interactive musical applications. Sending the proximity signal via *Open Sound Control* (OSC) allows any software which can receive OSC messages to map the interactions to musical parameters.

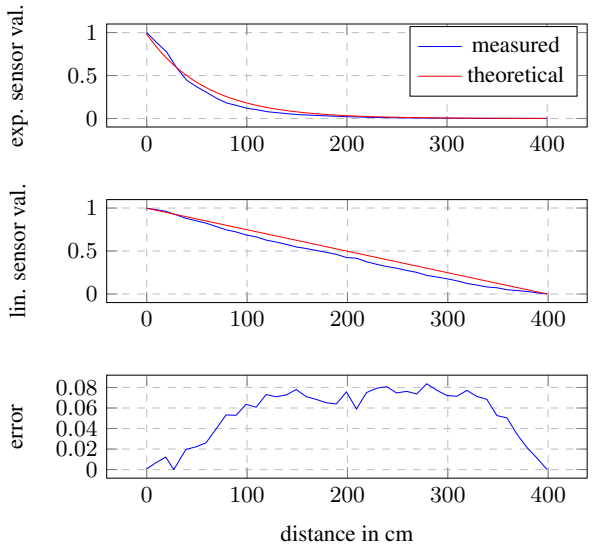


Figure 6. From top to bottom: Theoretical and measured exponential sensor values; theoretical linear values and measured values mapped from exponential to linear; error between theoretical values and measured values mapped to linear.

6. EVALUATION AND DISCUSSION

Within the interdisciplinary context of the research project, we developed new sensor technology for the sonification of proximity and touch. We built working and scalable wireless prototypes to be used in real-world and clinical research. The question of how interaction is facilitated through real-time auditory feedback is investigated by applying behavioural analysis using structured observation [37]. Our practical experience with Sentire in an waiting-list controlled trial in couple therapy emphasises the improved flexibility gained through the wireless version. Remaining undisturbed by cables attached to the body enables more freedom of movement; in addition, the less "visible" the technological system becomes for users, the more it may enable individuals to use it as therapeutic means for uncovering the intricacies of social interaction.

6.1 Performance

The sensor system can measure proximity up to approximately $4m$. Above this range, noise starts to shadow the measurement. Figure 6.1 shows proximity measurements in the range from $0cm$ to $400cm$, taken every $10cm$. The sensor values are normalised to the values at minimum ($0cm$) and maximum ($400cm$) distance. Furthermore, the figure compares the measured values—mapped from exponential to linear—with a theoretical linear trend and the resulting error. The mapping from exponential to linear could be optimised; however, qualitative tests indicate that more accurate curve fittings are rather hard to perceive in the sonification.

Based on the systematic investigation of different modes of inter-body coupling by Zhao et al. [31], we hypothesize that shifting to a frequency range above $10MHz$—the range of electro-magnetic coupling as opposed to electro-quasistatic coupling—would extend the possible range. Currently, we use an adaptive exponential lag filter to counterbalance the increasing noise in the far range. Although the sensing range of the wireless prototype suits the typical interactive space and is similar to the wired version of Sentire, shifting to a higher frequency domain would also make the system less susceptible to environmental noise; consequently, less filtering would be needed, and the system's reactivity could be improved. The filter has a lag time of $0.6s$ in the far range to $1ms$ for close distances. These settings are in line with our assumption that the system's reactivity is more relevant in the intimate space.

Since our proximity sensing is based on electro-quasistatic coupling, and the human bodies do not only couple with each other and the ground, but also with the environment, factors such as body size, floor material and nearby electrical devices can influence results. Nonetheless, the audible differences in the sonification are minor and not significantly relevant, as the users' perception is focused on changes in proximity rather than the precise measurement of physical distance and its representation in the sound environment. Since the contact surface slightly alters the signal strength when touching, a separate signal processing for values above the touch threshold could render a quantitative touch detection possible. However, at this stage of development, our software does not implement this feature yet.

6.2 Development Considerations

A central challenge in the hardware development was to overcome the need for a common ground connection as used in the wired version of Sentire. Our solution, custom-developed ground electrodes attached to shoe soles, required a considerable amount of product design and introduced the need to compensate for foot movement. While we have shown that using two independent prototypes per user and a dedicated algorithm can eliminate the influence of foot movement to a large extent, the solution doubles the technical overhead. Furthermore, the assumption that only one foot is raised at a time limits the compensation effect in real-case scenarios. Our latest iteration of foot electrodes based on SWNCTs as an additive to silicone served as proof-of-concept for manufacturing conductive electrodes that can be worn on shoe soles. However, these had to be rejected because of the unknown health risks of the materials.

Using a serial protocol for the transmission of the sensor signal to the base station is beneficial in terms of latency, but prone to losing packets and thus interferes with a stable proximity detection. However, because we will shift most of the sensor processing from SuperCollider to an embedded platform as part of the wearable, such interference is a minimal concern. The Teensy 3.2 microcontroller board and its software libraries turned out to be a pragmatic all-in-one solution for our evaluation boards, but future iterations should prioritise size and power efficiency and therefore implement a minimalistic solution in favour of wearability and usability. The current prototype iteration weighs approx. 320 grams in total (two times Electronics, Foot

Electrodes and Body Electrodes) and the main electronic parts (including the case, excluding the connectors) have a size of approx. $5cm \times 5cm \times 2.5cm$ (transceiver) and $5cm \times 5cm \times 1cm$ (signal amplifier) each. Battery life is mainly limited by the energy consumption of the signal amplifiers and, without any optimization, lasts for about 2 hours.

7. CONCLUSIONS AND OUTLOOK

We developed a new wireless sensor technology capable of measuring proximity and touch that overcomes the limitations of established systems for inter-body distance detection. The system has a range of approximately four metres and therefore covers the three relevant interpersonal zones as defined by Hall [1]: intimate, personal and social. Quantitative comparisons between the two systems will be the subject of our subsequent user study.

It is worth noting that, apart from closed-loop auditory interaction, the developed sensor technology could be used in a broad range of clinical and non-clinical applications. Furthermore, it is not limited to sensing proximity between individuals. Future usage could involve the detection of proximity between entities in general, including objects, provided they are conductive and sizable enough for affixing sensor electronics.

So far, a qualitative comparison shows the same effectiveness as the wired version of the system in couple therapy and, importantly, highlights the possibility of greater freedom in movement and thus improved usability.

Acknowledgments

We thank Terri Harel and Rebekka Gold for their proofreading and valuable comments. Moreover we thank Aminata Cisse for her student assistance in product design and Bruno Gola for his student assistance in software development. The Project was funded by the Federal Ministry of Education and Research (BMBF).

8. REFERENCES

[1] E. T. Hall, *The Hidden Dimension*. Garden City, N.Y.: Doubleday, 1966.

[2] S. Greenberg, N. Marquardt, T. Ballendat, R. Diaz-Marino, and M. Wang, "Proxemic interactions: The new ubicomp?" *Interactions*, vol. 18, no. 1, p. 42, Jan. 2011.

[3] J. Yang, T. Hermann, and R. Bresin, "Introduction to the special issue on interactive sonification," *Journal on Multimodal User Interfaces*, vol. 13, no. 3, pp. 151–153, Sep. 2019.

[4] E. Frid, L. Elblaus, and R. Bresin, "Interactive sonification of a fluid dance movement: An exploratory study," *Journal on Multimodal User Interfaces*, vol. 13, no. 3, pp. 181–189, Sep. 2019.

[5] T. Hermann, A. Neumann, and S. Zehe, "Head gesture sonification for supporting social interaction," in *Proceedings of the 7th Audio Mostly Conference: A Conference on Interaction with Sound*. Corfu, Greece: Association for Computing Machinery, Sep. 2012, pp. 82–89.

[6] M. Funk, K. Kuwabara, and M. J. Lyons, "Sonification of facial actions for musical expression," in *Proceedings of the 2005 Conference on New Interfaces for Musical Expression*. Vancouver, Canada: National University of Singapore, May 2005, pp. 127–131.

[7] G. Varni, G. Dubus, S. Oksanen, G. Volpe, M. Fabiani, R. Bresin, J. Kleimola, V. Välimäki, and A. Camurri, "Interactive sonification of synchronisation of motoric behaviour in social active listening to music with mobile devices," *Journal on Multimodal User Interfaces*, vol. 5, no. 3, pp. 157–173, May 2012.

[8] A. DeWitt and R. Bresin, "Sound Design for Affective Interaction," in *Affective Computing and Intelligent Interaction*, ser. Lecture Notes in Computer Science, A. C. R. Paiva, R. Prada, and R. W. Picard, Eds. Berlin, Heidelberg: Springer, 2007, pp. 523–533.

[9] J. H. Kim and U. Seifert, "Embodiment and Agency: Towards an Aesthetics of Interactive Performativity," in *4th Sound and Music Computing Conference*, Lefkada, Greece, Jan. 2007, pp. 230–237.

[10] R. Niewiadomski, M. Mancini, A. Cera, S. Piana, C. Canepa, and A. Camurri, "Does embodied training improve the recognition of mid-level expressive movement qualities sonification?" *Journal on Multimodal User Interfaces*, vol. 13, no. 3, pp. 191–203, Sep. 2019.

[11] T. Großhauser and T. Hermann, "Wearable Setup for Gesture and Motion based Closed Loop Audio Interaction," in *Proceedings of the 16th International Community for Auditory Display*. Washington D.C., USA: rish Centre for High End Computing, une 9–15, 2010.

[12] C. Erdem, K. H. Schia, and A. R. Jensenius, "Vrengt: A Shared Body-Machine Instrument for Music-Dance Performance," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Porto Alegre, Brazil, Jun. 2019, pp. 186–191.

[13] A. Müller, J. Fuchs, and K. Röpke, "Skintimacy: Exploring interpersonal boundaries through musical interactions," in *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*. Funchal, Portugal: Association for Computing Machinery, Jan. 2010, pp. 403–404.

[14] Design Research Lab, "Closer," n.d., [Online]. Available: https://www.drlab.org/research-projects/closer/. [Accessed: Feb. 8, 2022].

[15] T. Baba, T. Ushiama, and K. Tomimatsu, "Freqtric Drums: A Musical Instrument That Uses Skin Contact as an Interface," in *Proceedings of the 2007 Conference on New Interfaces for Musical Expression*, New York, USA, Jun. 2007, pp. 386–387.

[16] T. Hachisu and K. Suzuki, "Representing Interpersonal Touch Directions by Tactile Apparent Motion Using Smart Bracelets," *IEEE Transactions on Haptics*, vol. 12, no. 3, pp. 327–338, Jul. 2019.

[17] Sangli Design, "Expressive Wearable," 2014, [Online]. Available: https://www.sanglidesign.com/expressive-wearable. [Accessed: Feb. 8, 2022].

[18] S. Li, Y. Yang, M. He, Y. Wang, and C. Yao, "TONG: A wearable system to remind people of interpersonal distance," in *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*. London, United Kingdom: Association for Computing Machinery, Sep. 2019, pp. 117–120.

[19] S. Bian, B. Zhou, and P. Lukowicz, "Social Distance Monitor with a Wearable Magnetic Field Proximity Sensor," *Sensors*, vol. 20, no. 18, Sep. 2020.

[20] Metabody, "Touch Matters," n.d., [Online]. Available: https://metabody.eu/tools-for-understanding. [Accessed: Feb. 8, 2022].

[21] "Sentire: A deep sensory experience. human interaction transformed into sound," n.d., [Online]. Available: https://sentire.me/. [Accessed: Feb. 8, 2022].

[22] J. F. Zhao, X. M. Chen, B. D. Liang, and Q. X. Chen, "A Review on Human Body Communication: Signal Propagation Model, Communication Performance, and Experimental Issues," *Wireless Communications and Mobile Computing*, vol. 2017, Oct. 2017.

[23] T. Kang, K.-I. Oh, H. Park, and S. Kang, "Review of capacitive coupling human body communications based on digital transmission," *ICT Express*, vol. 2, no. 4, pp. 180–187, Dec. 2016.

[24] T. Zimmerman, J. Smith, J. Paradiso, D. Allport, and N. Gershenfeld, "Applying Electric Field Sensing to Human-Computer Interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., Aug. 1995, pp. 280–287.

[25] T. Grosse-Puppendahl, S. Herber, R. Wimmer, F. Englert, S. Beck, J. von Wilmsdorff, R. Wichert, and A. Kuijper, "Capacitive near-field communication for ubiquitous interaction and perception," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. Seattle, USA: Association for Computing Machinery, Sep. 2014, pp. 231–242.

[26] S. Bian, V. Rey, J. Younas, and P. Lukowicz, "Wrist-Worn Capacitive Sensor for Activity and Physical Collaboration Recognition," in *International Conference on Pervasive Computing and Communications*. IEEE, Mar. 2019, pp. 261–266.

[27] J. Li, Z. Nie, Y. Liu, L. Wang, and Y. Hao, "Evaluation of Propagation Characteristics Using the Human Body as an Antenna," *Sensors*, vol. 17, no. 12, p. 2878, Dec. 2017.

[28] B. Kibret, A. Teshome, and D. Lai, "Characterizing the Human Body as a Monopole Antenna," *IEEE Transactions on Antennas and Propagation*, vol. 63, pp. 4384–4392, Oct. 2015.

[29] A. Meharouech, J. Elias, and A. Mehaoua, "Moving Towards Body-to-Body Sensor Networks for Ubiquitous Applications: A Survey," *Journal of Sensor and Actuator Networks*, vol. 8, no. 2, p. 27, Mar. 2019.

[30] D. B. Arbia, M. M. Alam, Y. L. Moullec, and E. B. Hamida, "Communication Challenges in on-Body and Body-to-Body Wearable Wireless Networks—A Connectivity Perspective," *Technologies*, vol. 5, no. 3, p. 43, Jun. 2017.

[31] M. Nath, S. Maity, S. Avlani, S. Weigand, and S. Sen, "Inter-body coupling in electro-quasistatic human body communication: Theory and analysis of security and interference properties," *Scientific Reports*, vol. 11, no. 1, p. 4378, Feb. 2021.

[32] E. R. Miranda and M. M. Wanderley, *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*. Middleton, USA: A-R Editions, Inc., Jan. 2006.

[33] B. Stahl, M. Lussana, M. Rizzonelli, P. Staudt, A. Milek, and J. H. Kim, "Sentire: Exploring the Suitability of Movement and Sound in Couple Therapy," presented at Open Ground: Music Therapy in Collaboration & Exchange, London, 2021.

[34] P. Jackson, N. R. Jacobsen, A. Baun, R. Birkedal, D. Kühnel, K. A. Jensen, U. Vogel, and H. Wallin, "Bioaccumulation and ecotoxicity of carbon nanotubes," *Chemistry Central Journal*, vol. 7, no. 1, p. 154, Sep. 2013.

[35] C. Zhang, L. Wu, M. de Perrot, and X. Zhao, "Carbon Nanotubes: A Summary of Beneficial and Dangerous Aspects of an Increasingly Popular Group of Nanomaterials," *Frontiers in Oncology*, vol. 11, 2021.

[36] D. Livingstone and E. Miranda, "Composition for Ubiquitous Responsive Sound Environments," in *Proceedings of the International Computer Music Conference*, 2004, pp. 321–324.

[37] M. Rizzonelli, J. H. Kim, P. Staudt, and M. Lussana, "Fostering Social Interaction through Sound Feedback: Sentire," *Organised Sound (Upcoming)*, vol. 28, no. 01, pp. 317–326, 2022.

[38] J. Rovan, M. Wanderley, and S. Dubnov, "Instrumental Gestural Mapping Strategies as Expressivity Determinants in Computer Music Performance," in *Kansei, The Technology of Emotion. Proceedings of the AIMI International Workshop*. Genoa, Italy: Associazione di Informatica Musicale Italiana, Oct. 1997, pp. 68–73.

THE SPATBOX: AN INTUITIVE TRAJECTORY ENGINE TO SPATIALIZE SOUND

Pierre Lecomte
Univ Lyon, Univ Claude Bernard Lyon 1,
CNRS, Ecole Centrale de Lyon, INSA Lyon,
LMFA, UMR5509, 69622 Villeurbanne France
pierre.lecomte@univ-lyon1.fr

ABSTRACT

This paper presents the software part of the spatbox project [1] : an interface to intuitively generate 3D trajectories and to spatialize them in real-time on a loudspeaker array with Ambisonics. Each trajectory is described by a parametric curve, where each of the coordinates varies according to a configurable LFO. Depending on the LFO parameters, many trajectories can be generated. The similarity of the interface to that of a synthesizer allows the user to quickly get to grips with the tool. In particular, it is possible to create trajectory presets for fixed parameter configurations. An implementation in the Faust language is made and a visual feedback tool is introduced. Some additional features for the trajectory modification are proposed: choice of the coordinate system, trajectory scaling and acceleration, Doppler effect, hold/reset and rotation/translation functionalities.

1. INTRODUCTION

Spatial-, immersive-, or 3D-sound is now being widely spread to the greatest number. Notably in the cinema with commercial system such as Dolby Atmos [2] , or in the gaming sector with binaural sound. More recently, popular music concert are being offered with immersive sound for large venue, for instance with the L-ISA system [3] from L-Acoustics. However, the artistic creation of immersive sound content is still not easily accessible to the "ordinary" musician. This is mainly due to the fact that a lot of equipment is needed when making 3D sound: a dedicated room with many loudspeakers. Consequently, it is rather the professional musical industry or artistic creation centers that occupy this sector. Moreover, the composition often requires experimentation and reflection to choose the position of the sources in space and define their trajectories over time, although a live and improvised spatialization could open up a whole new creative horizon. From a software perspective, the offer is growing and many solutions allow professionals to create spatial sound content. One can mention software such as Sound Particles [4] , Ircam/Flux Spat Revolution [5] [1], GRM Tools [6] , Noise Makers [7] , Blue Ripple [8] , and many others [2]. On the open-source side: IEM-Plugin-suite [9] , ambitools [3] [10] or Sparta [11] [4] for sound generation and IanniX [12] for trajectory sequencing. These software fit well in a 3D audio content creation process with a Digital Audio Workstation (DAW) but are not directly designed for live spatialization where one wishes to improvise the sound trajectories. The use of hardware interfaces can be a relevant choice for this purpose. To the author's knowledge, there exist almost no autonomous interfaces that allow the spatialization of 3D sound on a loudspeaker array in an intuitive way [13] , apart from a mixing console used for the diffusion of sound, notably in the practice of electroacoustic music [5, 6].

It is in this context that the spatbox project takes place. It seeks to make an hardware autonomous interface which allows the user to instantly place sound on trajectories in space with a few loudspeakers, by turning a few knobs. With this instrument, the sounds travel on trajectories generated in an intuitive way, like creating sound patches on a synthesizer. As it is a kind of box to spatialize the sound, the project is called "spatbox". In this paper, the algorithms for generating the trajectories and computing the loudspeaker signals for the spatbox are described. An implementation in the Faust language [14] [7] is made and a software version of the spatbox is presented. The use of the Faust language allows to target an embedded system architecture for the hardware realization of the spatbox.

The paper proceeds as follows: In Sec. 2, the choice of the Ambisonics spatialization approach is argued, the needed equations are reviewed. The trajectory generation engine, two examples and several additional features are presented in Sec. 3. The user interface as well as a visual feedback tool to view the trajectories are introduced

[1] http://www.sekisushai.net/spatbox/
[2] https://www.dolby.com/technologies/dolby-atmos/
[3] https://www.l-isa-immersive.com/

Copyright: © 2022 Pierre Lecomte. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[4] https://soundparticles.com/
[5] https://www.flux.audio/project/spat-revolution/
[6] https://inagrm.com/fr/store
[7] https://www.noisemakers.fr/
[8] https://www.blueripplesound.com/
[9] https://plugins.iem.at/
[10] http://www.sekisushai.net/ambitools/
[11] https://leomccormack.github.io/sparta-site/
[12] https://www.iannix.org/en/
[13] https://blog.bela.io/multichannel-sound-spatialisation/
[14] https://faust.grame.fr/

in Sec.4. Finally, conclusion and future works are done in Sec. 5.

2. AMBISONICS

There exists many approaches to spatialize sound in space with a loudspeaker array. Among the most popular are Vector Base Amplitude Panning (VPAB) [8], Wave Field Synthesis (WFS) [9] or Higher Order Ambisonics [10], simply called Ambisonics for the rest of this paper. The trajectory engine described in Sec. 3 could be independent of the spatialization technique. However, the main objective of the spatbox is to provide an autonomous interface that takes the sound signals to be spatialized and as inputs and outputs the loudspeaker signals. Therefore the spatialization engine should be embedded.

The choice is made to use Ambisonics in this project for several reasons. First, the Ambisonics signals flow goes through encoding, transformation (i.e. spatial audio effects) and decoding of the sound field [10, Chap. 5]. For the encoding step, it is possible to encode the source distance using near-field filters [11], which is an key feature for a realistic rendering of 3D trajectories. Then, spatial transformations are possible before the decoding step, for example reverberation, spatial blur [12] or warping [13] of the sound scene. These effects can then be integrated into the spatbox spatialization engine with dedicated controllers, as a master effect on a mixing console. Finally, the decoding of the sound scene is flexible and can be done on an irregular loudspeaker array [14] or even in binaural for a headphone rendering [15]. The control of the spatial resolution is easily done with the degree of decomposition on the basis of spherical harmonics and immersive sound results can be obtained with only less than a dozen loudspeakers. Thus, it becomes possible to realize a hardware interface requiring few loudspeakers and thus makes the project accessible to the largest number of musicians.

In the rest of this section, one recalls the main Ambisonic equations which are used to encode, decode or directly pan a virtual source on a loudspeaker array. For a more extensive description, refer to [16] for example.

2.1 Point Source Encoding

Let's consider a point source, located at a position $(r_s(t),\ \theta_s(t),\ \phi_s(t))$ in the spherical coordinate system, where $r \in]0,\ \infty[$ is the radius, $\theta \in]-\pi,\ \pi]$ the azimuth angle, $\phi \in [-\pi/2,\ \pi/2]$ the elevation angle and t represents the time variable. This source carries a signal $s(t)$. The ambisonic encoding of this source produces the signals $b_{l,m}(t)$ of degree l, order m with $(l,m) \in (\mathbb{N} \times \mathbb{Z}), |m| \leq l$, such that:

$$b_{l,m}(t) = s(t - r_s/c) * \text{NF}_l(r_s(t), r_0, t) \frac{r_0}{r_s} Y_{l,m}(\theta_s(t), \phi_s(t)). \quad (1)$$

In Equation (1), $c \simeq 340$ m/s is the sound speed, $Y_{l,m}$ is the real spherical harmonic of degree l and order m, $*$ is the convolution operator and NF_l are the near-field filters of degree l [11]. In the current implementation, these filters are stabilized by Near-Field Compensation (NFC) filters at a fixed radius $r_0 = 1$ m [3]. When $r_s < r_0$, a "bass-boost" effect occurs. To limit the latter, which is combined with an amplification in $1/r_s$ in Equation (1), a minimum radius $r_{\text{min}} = 0.5$ m is introduced as the minimal possible distance for the source. For an encoding up to degree L, the corresponding maximal amplification of the $s(t)$ is $(L+1) \times 20 \log_{10}(r_0/r_{\text{min}}) \simeq 6(L+1)$ dB.

2.2 Sampling Ambisonic Decoder

The decoding step on a loudspeaker array consists in calculating the signals of the loudspeakers from the ambisonic signals $b_{l,m}$. Many strategies exist for this step. A review can be found in [10, chapt. 4] for example.

In this paper one uses the Sampling Ambisonic Decoder (SAD) technique. The loudspeaker signals are obtained from the simple source continuous formulation [16] sampled in the direction of the loudspeakers as:

$$g_n(t) = \frac{r_n}{r_{\text{max}}} \sum_{l=0}^{L} w_{\text{max-}\mathbf{r}_E,l,L} \text{NFC}_l(r_n, t) * \sum_{m=-l}^{l} Y_{l,m}(\theta_n, \phi_n) b_{l,m} \left(t - \frac{r_{\text{max}} - r_n}{c} \right). \quad (2)$$

In Equation (2), n stands for the $n-$th loudspeaker in an array of N loudspeakers. The ambisonics signals are weighted with max-$\mathbf{r}_E$ weights, denoted $w_{\text{max-}\mathbf{r}_E,l,L}$ in order to maximize the energy vector $\mathbf{r}_E$, improving high frequency localization [17]. Note that the near-field effect of each loudspeaker is compensated with NFC filters. Moreover, the amplitude is scaled to r_n/r_{max} where r_{max} is the maximal distance among the N loudspeakers and the signal is delayed by the corresponding propagation distance relative to r_{max}. This ensures that all loudspeakers have the same gain and delay at origin when playing the same signal. The choice of the decomposition degree L (i.e. Ambisonic order), is generally related to the number of loudspeaker N, such that $N \geq (L+1)^2$.

2.3 Equivalent Panning Law

By combining Equations (1) and (2), and using the addition theorem of spherical harmonics, an equivalent panning law is derived [16] as:

$$g_n(t) = s \left(t - \frac{r_s + r_{\text{max}} - r_n}{c} \right) \frac{r_n}{r_{\text{max}}} \frac{r_0}{r_s} \sum_{l=0}^{L} (2l+1) w_{\text{max-}\mathbf{r}_E,l,L} * \text{NFC}_l(r_n, t) * \text{NF}_l(r_s(t), r_0, t) P_l(\cos(\gamma_n(t))), \quad (3)$$

where P_l is the l-th Legendre polynomial, and $\gamma_n(t) = \cos(\phi_s(t)) \cos(\phi_n) \cos(\theta_s(t) - \theta_n) + \sin(\phi_s(t)) \sin(\phi_n)$ is the angle between the source and the n-th loudspeaker. This formulation allows obtaining the loudspeaker signals with much fewer computation than the formalism with encoding and decoding steps. This is a good point for efficient implementation, especially when making the autonomous hardware interface by using an embedded system with low computing power. However the use of such

panning law no longer allows the use of spatial ambisonic effects before decoding. In addition, if the loudspeakers are not arranged in a "regular way" around the origin, reduced loudness at the location of poor loudspeaker coverage will be noticeable [10, p. 72].

3. TRAJECTORY ENGINE

The aim of the spatbox trajectory engine is to generate the signals $r_s(t)$, $\theta_s(t)$ and $\phi_s(t)$ in the Eqs (1) or (3), i.e. the trajectory that the source follows over time. There are an infinite number of possible trajectories and as many equations to describe them. One therefore constrain this problem in the following way:

- The trajectories are described in a parametric way so that the user can generate a multitude of them by varying the parameters.
- A maximum of trajectories should be generated with a minimum of parameters. Indeed, as the final goal is to make an hardware interface, the minimum number of controllers are desired.

The problem is then to find how to describe the trajectories in a parametric way and which parameters to associate to them. A proposal for a solution to this problem is made in the following Sec. 3.2. The current implementation of this engine is made with the Faust language. The ambisonic layer is made with ambitools[10] [3].

3.1 Parametric Equation

Let be a trajectory that represents all the positions ($r_s(t)$, $\theta_s(t)$, $\phi_s(t)$) that the source takes over time. This trajectory is expressed in a parametric way as follows, where the parameter is the time t:

$$\begin{aligned} r_s(t) &= \sqrt{x_s(t)^2 + y_s(t)^2 + z_s(t)^2} \\ \theta_s(t) &= \arctan(y_s(t)/x_s(t)) \\ \phi_s(t) &= \arcsin(z_s(t)/r_s(t)) \end{aligned} \tag{4}$$

In Equation (4), the functions $x_s(t)$, $y_s(t)$ and $z_s(t)$ are the trajectory coordinates functions in the Cartesian coordinate system. The choice of the coordinate system is left to the user for each trajectory to generate. Thus, either the signals, $r_s(t)$, $\theta_s(t)$, $\phi_s(t)$ in spherical coordinates are directly generated, or the signals $x_s(t)$, $y_s(t)$, $z_s(t)$ in Cartesian coordinates are generated and then converted with Equation (4).

3.2 Low Frequency Oscillators as Trajectory Coordinates Functions

To express the time variations of ($x_s(t)$, $y_s(t)$, $z_s(t)$) or ($r_s(t)$, $\theta_s(t)$, $\phi_s(t)$), i.e. the trajectory coordinates functions in Equation (4), one chooses to use Low Frequency Oscillators (LFO)s. LFOs are commonly used in synthesizers to modulate the parameters of the sound patch. Used here as coordinate generators, they can produce a multitude of different trajectories in three dimensions. A LFO is usually a periodic function that oscillates around 0 and that can be characterized by its frequency f, waveform, amplitude a, phase or duty cycle p:

$$\text{LFO}(t) = a \times \text{waveform}(f, p, t) \tag{5}$$

The phase/duty cycle $p \in [0\ 1]$ is set as a fraction of the waveform unit period. The frequency f of the LFO is kept low as it represents the cyclic variations of the corresponding coordinate signal, and high speed can be reached with only a few Hertz. In the current implementation, $f \in [0\ 1]$ Hz, and the user sets one LFO per coordinate : $\text{LFO}_{x_s}(t)$, $\text{LFO}_{y_s}(t)$, $\text{LFO}_{z_s}(t)$ or $\text{LFO}_{r_s}(t)$, $\text{LFO}_{\theta_s}(t)$, $\text{LFO}_{\phi_s}(t)$.

Bounds When assigned to the Cartesian coordinates $\text{LFO}_{x_s}(t)$, $\text{LFO}_{y_s}(t)$ or $\text{LFO}_{z_s}(t) \in [-a,\ a]$ with $a \in [0,\ 1]$. When assigned to the spherical coordinates, $\text{LFO}_{r_s}(t) \in [-a + 1 + r_{\text{min}},\ a + 1 + r_{\text{min}}]$, $\text{LFO}_{\theta_s}(t) \in [-a\pi,\ a\pi]$ and $\text{LFO}_{\phi_s}(t) \in [-a\pi/2\ , a\pi/2]$.

Waveforms Six different waveforms are proposed in the current implementation, as shown in Tab. 1 over two periods. Note that for the square wave, the user sets the duty cycle instead of the phase. The noise waveform is generated with a sample-and-hold technique where a sample is hold during one period. The phase parameter in this situation sets the instant during the period when a new sample is hold.

ID	Name	Waveform
0	Sawtooth	
1	Sawtooth 2	
2	Sine	
3	Triangle	
4	Square	
5	Noise	

Table 1. The six waveforms for the LFOs used as trajectory coordinates functions. The waveform are shown over two periods.

3.3 Trajectory Generation Principle

To generate a trajectory patch, one proceeds as follows (see Fig. 3 for a graphical user interface view). First, one chooses the coordinate system between Cartesian or spherical. Then one sets each LFO waveform for each coordinates and finally one sets the LFO parameters : amplitude, frequency, phase. Following this procedure, two examples of trajectory are given:

Helix In Cartesian coordinates, a parametric Helix trajectory around the $z-$axis can be expressed with LFOs as follows:

$$\begin{aligned} x_s(t) &= a_0 \sin(2\pi f_0 t + p_0) \\ y_s(t) &= a_1 \sin(2\pi(f_0 t + p_0 + 0.25)) \\ z_s(t) &= a_2 \text{ sawtooth}(f_2, p, t) \end{aligned} \tag{6}$$

The first $\text{LFO}_{x_s}(t)$ is a sinewave with amplitude a_0, frequency f_0 and phase p_0, The second $\text{LFO}_{y_s}(t)$ is a cosine wave with respect to $x_s(t)$, obtained from a sinewave waveform where the phase is set to $p_1 = p_0 + 0.25$. Its amplitude is a_1. With $f_1 = f_0$, the horizontal trajectory is an ellipse of semi-axis a_0 and a_1. For the third $\text{LFO}_{z_s}(t)$, a sawtooth waveform of frequency f_2, amplitude a_2, phase p_2 sets the pitch of the Helix. This trajectory is visible in red in Fig. 1 and the corresponding patch parameters are shown on the left side of Fig. 3.

Flower In spherical coordinates, a flower-shaped trajectory centered at origin in the horizontal plane can be described as follows:

$$\begin{aligned} r_s(t) &= a_0(\sin(2\pi(f_0 t + p_0)) + 1) + r_{\text{min}} \\ \theta_s(t) &= a_1 \text{sawtooth}(2\pi f_1 t + p_1) \\ \phi_s(t) &= 0 \end{aligned} \tag{7}$$

The first $\text{LFO}_{r_s}(t)$, produces a sinusoidal variation of $r_s(t)$ with minimal radius r_{min}, while the second $\text{LFO}_{\theta_s}(t)$ is a sawtooth, which produces a linear increase in azimuth angle. The third $\text{LFO}_{\phi_s}(t)$ is set to null amplitude and phase such that the trajectory stays on the horizontal plane. This trajectory is visible in cyan in Fig. 1 and the corresponding patch parameters are shown on the right side of Fig. 3.

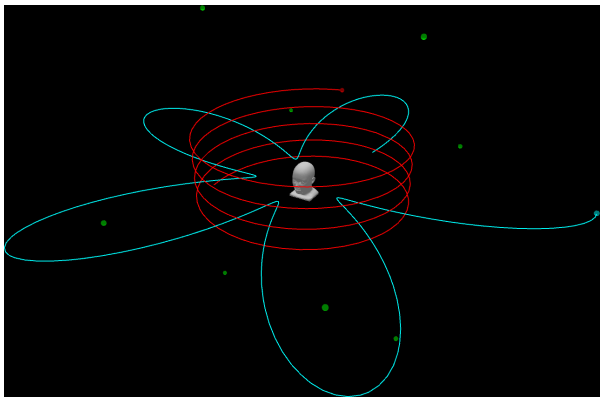


Figure 1. Example of two trajectories generated with the spatbox and visualized on the 3D visualizer: A helix in red and a flower-shaped trajectory in cyan. The green points correspond to the position of the loudspeakers.

Through these two simple examples, one can realize the great variety of trajectory patches that can be made by varying waveforms, frequency, amplitude and phase ratio between each LFOs. Since LFOs are periodic functions (except for the "noise" waveform), the resulting trajectories are also periodic as long as the patch parameters are not moved. As a result, they could fit well with a repetitive sound signal as found in electronic music for instance. Note however that it is quite simple to make the trajectory evolve over a long period of time or to modulate it, by slightly varying a frequency ratio between the LFOs.

3.4 LFOs Synchronization

One can notice, from the Helix trajectory example, that the phase relationship between the LFOs is critical to define the trajectory properly. Since the phase at the origin is defined independently for each LFO, it is crucial to keep a synchronization between the LFOs when the frequency or phase parameters of one of them is changed. To achieve this, in the current implementation, as soon as one of these parameters moves, all LFOs are reset to $t = 0$ to keep their synchronization. Consequently the source returns to the beginning of the trajectory while the user is changing the monitored parameters. Note that a very fast random position change effect can be produced in this way, using the noise waveform: Indeed, as the LFO is reset, a new value for the noise is produced, which results in an immediate change of the assigned coordinates.

3.5 Forbidden Sphere

As explained in Sec. 2.1, a minimum radius is introduced, r_{min}, below which the source cannot go, to avoid excessive signal amplification (or even a singularity if $r_s = 0$). Therefore, the signal $r_s(t)$ is compared to r_{min} and equalized to it if found to be smaller. Consequently, a forbidden sphere of radius r_{min} is defined and it surrounds the origin. The source can never enter inside it. If the trajectory encounters this sphere, it "slides" over it as $\theta_s(t)$ and $\phi_s(t)$ continue to evolve. An example of such situation is shown in Fig. 2.

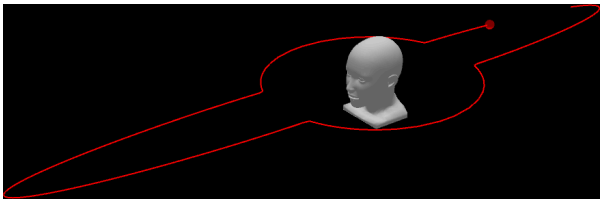


Figure 2. Example of an elliptic trajectory which encounters the forbidden sphere. As the radius $r_s(t) \geq r_{\text{min}}$ for all t, the source slides over the forbidden sphere.

3.6 Extra Features

This section presents some additional features to interact with the sound or the trajectories generated by the LFOs.

Doppler Effect A delay, which corresponds to the propagation time r_s/c to the origin (see Equations (1) and (3)) is eventually applied to the signal $s(t)$. When the source moves, this delay also varies and can produce a Doppler

effect. This results in changing the pitch of the signal $s(t)$, which can be an unwanted effect. Therefore, the application of this delay is left as an option to the user. Note that a playability problem may occur if the propagation delay is too large. For example, if the user places the source at 10 m while playing on a keyboard, it will take about 29 ms before the sound is heard.

Hold and Reset A hold function allows to stop the oscillation of the LFOs and to maintain them at their current value. Consequently, the source pauses on its trajectory. As soon as the hold is released, the source continues moving on its trajectory. In the same way, a reset function brings the source back to the beginning of its trajectory when activated.

Scale and Speed The amplitude of the LFOs is chosen by the user with a parameter $a \in [0\ 1]$ as mentioned in Sec. 3.2. A scale function allows to multiply the LFOs amplitude by an additional scaling factor. This scales up the whole trajectory. For Cartesian coordinates this function is realized by multiplying each LFO by the scale factor. For spherical coordinates, only the LFO assigned to $r_s(t)$ is multiplied by this scale factor. In the same way, a speed function can multiply each LFO frequency by a factor, resulting in an acceleration of the source on its trajectory.

Rotation and Translation The coordinate system in which the trajectories are expressed is centered on the origin. However, a rotation feature is available and allows to rotate the coordinate system around the z, y and x axes with yaw, pitch and roll angles respectively. As well, a translation can be made along the x, y and z axes. As a result, the trajectories can be rotated and translated.

4. USER INTERFACE

This section introduces the spatbox software user interface.

4.1 Graphical User Interface

In its current version, the spatbox is only available as a software, either as a standalone tool or as a plugin, depending on the architecture targeted during the Faust compilation. The number of source to spatialize, it set for the compilation. An example of graphical user interface for 2 sources is visible on Fig. 3 for a Linux Alsa standalone application. For each source, 3 menus, 17 rotary knobs, 2 buttons and 1 checkbox allow creating a trajectory patch. The Doppler effect can be activated and the sound volume be adjusted with a check box and a gain knob respectively. Note that it is to be expected that the volume of the source will naturally increase or decrease as the source moves closer or further away. The controllers labels “0”, “1” and “2” correspond to the coordinates x, y, z or r, θ ϕ respectively, depending on the choice of the coordinate system between Cartesian of spherical. For the LFO bank menu, the correspondence between the identification number and the waveform is shown in Tab. 1. Meters of the coordinate signals allow seeing the current value of each coordinate in the spherical system. The values of these meters are transmitted by Open Sound Control (OSC) message to the 3D visualizer presented in the following section.

4.2 Visual Feedback

Although trajectory patch generation can be intuitive, it is very difficult to rely solely on one’s auditory system to accurately localize the position of sources and monitor the trajectories. Indeed, in addition to the limitation of human spatial perception, the localization performances can be degraded depending on the ambisonic resolution and the loudspeaker array. See for example the listening tests reported in [10] for further reading on this topic. Therefore, it seems essential for the user to have a visual feedback to monitor the trajectories he produces. The latter is made with the Processing [15] language [3]. An illustration is shown in Figs. 1 and 2. It consists of a 3D view of a dummy’s head, centered at the origin and looking in the direction of positive x-axis. The proportions are preserved and the dimensions are in meters. The loudspeakers are represented by green dots whose size and color can change depending on the signal level in dBfs. The successive positions of the sources are received via OSC messages by the Faust application presented in Sec. 4.1. They are kept in a First In First Out (FIFO) memory buffer which keeps only the last 500 positions (this value can be changed). The trajectories are drawn by connecting the successive positions with lines. The current position of the source is materialized by a colored ball. The user can move, zoom and pan the camera to improve its perspective visualization.

5. CONCLUSION AND FUTURE WORKS

In this paper, the spatbox project was introduced as a tool for intuitive spatialization on a loudspeaker array. The LFO-based trajectory generation engine was presented along with several additional features to interact with the sound and the trajectories. The periodic character of the generated trajectories leads to the idea of synchronizing the frequency of the LFOs to a metronome, for example a MIDI clock. This is the topic of future work. An implementation in the Faust language has been made as well as a visual feedback tool to observe the trajectories. Currently, the spatbox exists only in a software form although a MIDI interface can already be connected to associate physical controllers to the interface parameters. A hardware version is currently being designed and built. It targets an embedded system platform such as Teensy [16], Bela [17] or Raspberry Pi [18]. On these platforms, recent audio interfaces are getting multichannel: up to 6 inputs and 8 outputs on the Teensy and Rasberrypi and even up to 16 outputs on the Bela. Thus, this will allow the spatbox to become a musical instrument on its own, capable of receiving audio signals as input and spatializing them in real time on a array of up to 16 loudspeakers. The visual feedback tool seems

[15] https://processing.org/
[16] https://www.pjrc.com/teensy/
[17] https://bela.io/
[18] https://www.raspberrypi.com/

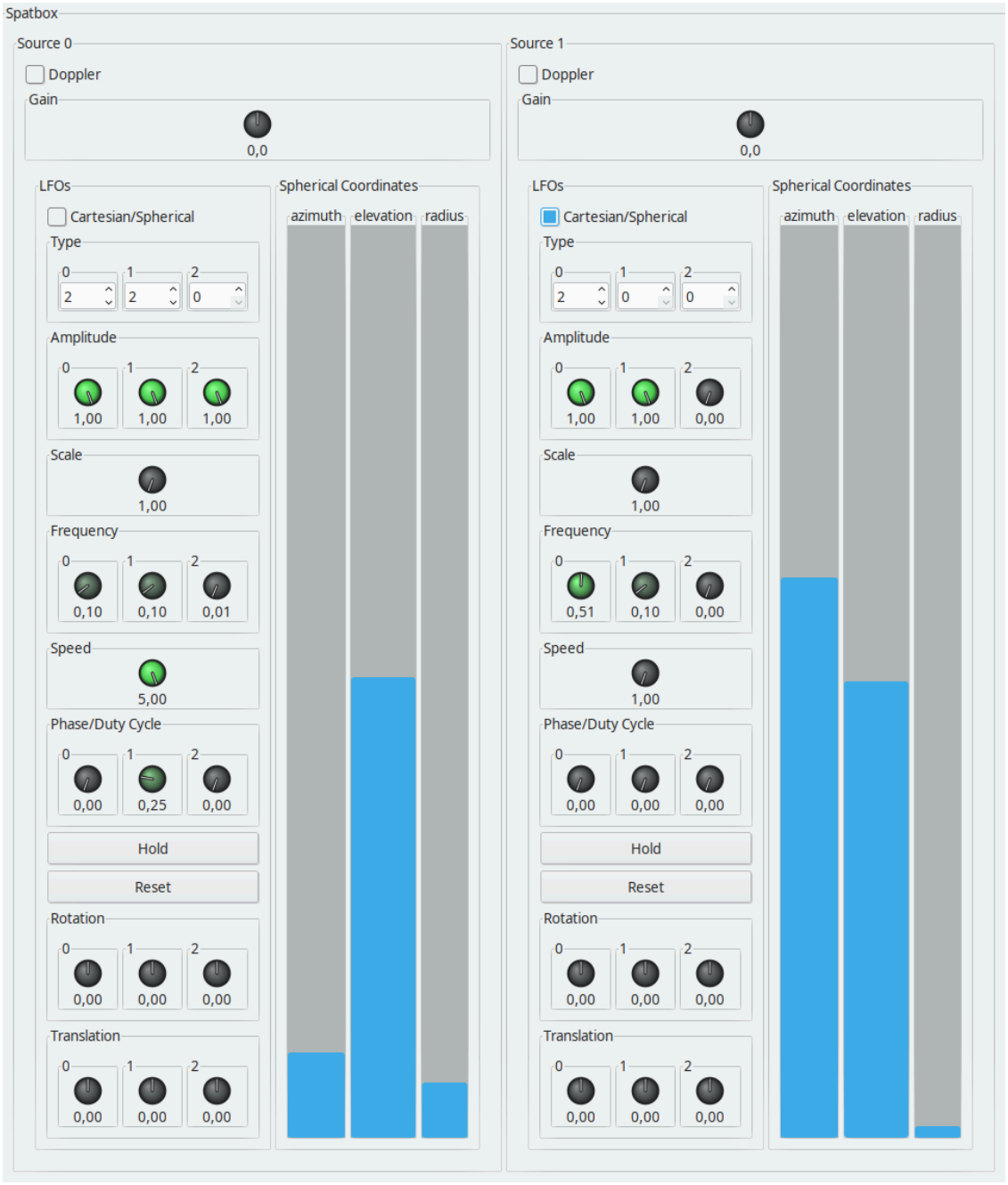


Figure 3. Graphical user interface for a standalone Linux Alsa application. Two sources are spatialized and their trajectory patch is set with the different controllers. The corresponding trajectories are visible on Fig. 1: The red trajectory corresponds to Source 0 and the cyan trajectory to Source 1.

however essential to the user and an embedded version of it is currently under study, although it is already possible to drive it on a computer with OSC messages. Once the hardware interface is available, it is planned to conduct a feedback study with musicians. Code and documentation for this project are available online [1] .

6. REFERENCES

[1] T. Carpentier, M. Noisternig, and O. Warusfel, "Twenty years of Ircam Spat: Looking back, looking forward," in *41st International Computer Music Conference (ICMC)*, 2015, pp. 270–277.

[2] J. D. Mathew, S. Huot, and B. F. Katz, "Survey and implications for the design of new 3D audio production and authoring tools," *Journal on Multimodal User Interfaces*, vol. 11, no. 3, pp. 277–287, 2017.

[3] P. Lecomte, "Ambitools: Tools for Sound Field Synthesis with Higher Order Ambisonics - V1.0," in *International Faust Conference*, Mainz, 2018, pp. 1–9.

[4] L. McCormack and A. Politis, "SPARTA & COMPASS: Real-time implementations of linear and parametric spatial audio reproduction and processing methods," in *Audio Engineering Society Conference: 2019 AES International Conference on Immersive and Interactive Audio*. York: AES, 2019, pp. 1–12.

[5] J. Harrison, "Sound, space, sculpture: Some thoughts on the 'what','how'and 'why'of sound diffusion," *Organised Sound*, vol. 3, no. 2, pp. 117–127, 1998.

[6] J. Prager, *L' Interprétation Acousmatique - Fondements Artistiques et Techniques de l'interprétation Des Œuvres Acousmatiques En Concert (in French)*, unpublished ed., 2012.

[7] Y. Orlarey, D. Fober, and S. Letz, "FAUST: An efficient functional approach to DSP programming," *New Computational Paradigms for Computer Music*, vol. 290, 2009.

[8] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," *Journal of the Audio Engineering Society*, vol. 45, no. 6, pp. 456–466, 1997.

[9] A. J. Berkhout, D. de Vries, and P. Vogel, "Acoustic control by wave field synthesis," *The Journal of the Acoustical Society of America*, vol. 93, no. 5, pp. 2764–2778, 1993.

[10] F. Zotter and M. Frank, *Ambisonics - A Practical 3D Audio Theory for Recording, Studio Production, Sound Reinforcement, and Virtual Reality*, ser. Springer Topics in Signal Processing. Cham, Switzerland: Springer, 2019.

[11] J. Daniel, "Spatial sound encoding including near field effect: Introducing distance coding filters and a viable, new ambisonic format," in *Audio Engineering Society Conference: 23rd International Conference: Signal Processing in Audio Recording and Reproduction*. Helsingør: AES, 2003, pp. 1–15.

[12] T. Carpentier, "Ambisonic spatial blur," in *Audio Engineering Society Convention 142*. Berlin: AES, 2017, pp. 1–7.

[13] H. Pomberger and F. Zotter, "Warping of 3D ambisonic recordings," in *Proc. of the 3rd Int. Symp. on Ambisonics & Spherical Acoustics*, Lexington, 2011.

[14] F. Zotter and M. Frank, "All-round ambisonic panning and decoding," *Journal of the Audio Engineering Society*, vol. 60, no. 10, pp. 807–820, 2012.

[15] M. Noisternig, A. Sontacchi, T. Musil, and R. Holdrich, "A 3d ambisonic based binaural sound reproduction system," in *Audio Engineering Society Conference: 24th International Conference: Multichannel Audio, The New Reality*. AES, 2003, pp. 1–5.

[16] P. Lecomte, P.-A. Gauthier, C. Langrenne, A. Berry, and A. Garcia, "A Fifty-Node Lebedev Grid and Its Applications to Ambisonics," *Journal of the Audio Engineering Society*, vol. 64, no. 11, pp. 868–881, 2016.

[17] J. Daniel, J.-B. Rault, and J.-D. Polack, "Ambisonics encoding of other audio formats for multiple listening conditions," in *Audio Engineering Society Convention 105*. San Francisco: AES, 1998, pp. 1–29.

A Gaze-Driven Digital Interface for Musical Expression Based on Real-time Physical Modelling Synthesis

Devansh Kandpal
Aalborg University
dkandp20@student.aau.dk

Prithvi Ravi Kantan
Aalborg University
prka@create.aau.dk

Stefania Serafin
Aalborg University
sts@create.aau.dk

ABSTRACT

Individuals with severely limited physical function such as Amyotrophic Lateral Sclerosis (ALS) sufferers are unable to engage in conventional music-making activities, as their bodily capabilities are often limited to eye movements. The rise of modern eye-tracking cameras has led to the development of augmented digital interfaces that can allow these individuals to compose music using only their gaze. This paper presents a gaze-controlled digital interface for musical expression and performance using a real-time physical model of a xylophone. The interface was designed to work with a basic Tobii eye-tracker and a scalable, open-source framework was built using the JUCE programming environment. A usability evaluation was carried out with nine convenience-sampled participants. Whilst the interface was found to be a feasible means for gaze-driven music performance our qualitative results indicate that the utility of the interface can be enhanced by expanding the possibilities for expressive control over the physical model. Potential improvements include a more robust gaze calibration method, as well as a redesigned graphical interface with more expressive possibilities. Overall, we see this work as a step towards accessible and inclusive musical performance interfaces for those with major physical limitations.

1. INTRODUCTION

Expression is a central component of human behaviour, and music is one of the oldest and most fundamental forms of expression [1]. Both conventional and *digital musical interfaces (DMIs)* and instruments [2] contribute strongly to human expression and interaction. Most forms of musical expression require some form of bodily movement, which means that individuals suffering from debilitating movement disorders can be severely limited in this regard. Advances in modern music technology have made it possible to address this through DMIs, and in recent years, many DMIs have been developed for individuals with physical limitations [3]. These have been termed as *adaptive digital musical instruments (ADMIs)*.

Copyright: © 2022 Devansh Kandpal et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

In a recent systematic review [4], ADMIs were segregated into categories including *tangible, touchless, biocontrollers, wearable,* etc., with tangible and touchless (camera-based) interfaces accounting for the majority. Of the remainder, there were only two gaze-controlled interfaces. Given the incredible communicative power of human eye and its role in regulating human interaction and emotional connection (and the advent of inexpensive eye-tracking technology) [5], it is interesting that the potential of gaze as an expressive medium has not been explored to a greater degree. Eye-driven ADMIs could, for instance, be invaluable to individuals suffering from ALS [6], who lose movement in all their limbs and are left with a very limited palette of interactive gestures. Moreover, existing work has proved that it is feasible to control sequencers and melodies using gaze [7, 8] to produce expressive performances. A gaze-based MIDI controller has also been developed [9] for use with digital audio workstations such as Ableton Live.

We believe that much of the potential of gaze-based ADMIs remains untapped. For instance, existing systems have typically used pre-recorded samples or simple oscillators for sound generation. Modern computing hardware has made it possible to manipulate more realistic-sounding physical models in real-time, which have been applied in interactive sonification research [10]. Physical models are particularly interesting for gaze-controlled ADMIs simply due to the sheer amount of parametric control that can be exercised over their sonic properties by a multitude of gaze parameters [5] with minimal physical effort. Current gaze-based DMIs are yet to explore modalities for interaction with real-time physical models, and to the best of our knowledge, there are no existing technological frameworks that integrate robust eye-tracking with physical modelling synthesis. Such a framework would greatly facilitate interdisciplinary and iterative user-centered studies, an indispensable component of human-computer interaction research [11].

The goal of the present work was to build a scalable and robust prototype framework to serve as a tool for exploring the interactive potential of gaze-controlled physical models. Subsequent sections cover the state-of-the-art in eye tracking and gaze-based DMIs, followed by the implementation, and usability evaluation of our prototype.

2. RELATED RESEARCH

The phrase *eye tracking* typically refers to the process of localizing the instantaneous gaze of a user. Gaze data is

classified into two categories, fixations and saccades [12]. Fixations are definite, fixed gazes of a user at a certain spot, and saccades are rapid movements between two points that may also be involuntary [12]. Dwell time, which essentially is a measure of how long the gaze of a user dwells on a particular point, is a well known parameter that is used to differentiate between fixations and saccades [12]. In terms of design, eye trackers may be head-mounted or remote, and their working mechanism may be video-based tracking, infrared pupil-corneal reflection tracking, or based on electrooculography [5].

Gaze-based ADMIs have used both head-mounted [9] and remote [7, 8] eye-tracking systems. Aside from requiring accurate and regular calibration [5], fixation detection and smoothing algorithms have been found to be necessary for consistent system behavior [7]. The interaction itself has typically taken place on a graphical interface, with the gaze point serving as a *cursor* that manipulates visual elements. Interface design is constrained by limitations in the spatial and temporal resolution of eye-tracking systems [5]. Some existing guidelines are that on-screen artifacts must not be very small in size, and should be spaced out adequately and evenly [7, 13]. Other known ADMI design considerations relate to latency, inconsistent spatial accuracy, and the aptly-named Midas touch problem [8] where all viewed objects are selected [5].

From an interaction perspective, gaze-based ADMIs have provided control over step sequencers [7, 8], MIDI parameters [9], and melody notes [7]. However, the lack of time- and space-accurate parallel control over GUI elements translates to increased interface complexity for achieving standard music production tasks. In the task of playing a timed melody, the manipulation of timed sequencers addresses the difficulty of navigating quickly and accurately to on-screen note triggers, but adds a large number of interface elements. Note triggers also need to be arranged in a different manner than traditional instruments (e.g, in a ring [7, 13]), potentially making them less intuitive. In the only documented rigorous evaluation of a gaze-based ADMI, EyeHarp users [7] experienced steep learning curves and comparable difficulty in mastering the ADMI to a traditional musical instrument. The amount of expressive potential in existing ADMIs is also limited due to the absence of expressive integrated synthesizers; only simple oscillators and MIDI note triggers have been used [7–9], and more complex algorithms have yet to be integrated and experimented with.

Overall, gaze-based ADMIs are still at an early stage, and considerable advances are necessary in order to fully unlock the expressive potential of human gaze. An interesting possibility is that of integrating online gaze tracking with real-time physical modelling synthesis, a technique with the potential to generate realistic sounds, albeit at a relatively high computational cost [14]. We see this as a hugely promising direction as it can help achieve a potentially vast span of compelling and expressive parameter mappings to link gaze and musical sound. At this point, it is unclear whether a robust and usable implementation can be achieved for real-time operation. Therefore, the primary aim of the present work was to build a scalable prototype to (A) evaluate the feasibility of combining gaze tracking and physical modelling synthesis in a real-time interface, and (B) serve as an open-source framework to facilitate the future development of gaze-based ADMIs.

3. DESIGN AND IMPLEMENTATION

We developed a system integrating eye tracking, a gaze-controlled graphical interface and physical modelling synthesis into one software application. This section details the interaction design and implementation specifics. We used the following tools during the development process:

1. **MATLAB:** This was used to prototype the physical model of a single xylophone bar and tune the subsequent bars.

2. **JUCE:** For our software application, we used the JUCE framework for the real-time audio processing and GUI classes it provided.

3. **Tobii X2-30 Eye Tracker:** This model is of the remote variety, allowing for unencumbered head movement. We chose this for its simple interfacing with a windows laptop using the Tobii Eye Tracker Manager software, which allows user-friendly calibration of the device. The software was interfaced with JUCE using the C-based Tobii SDK.

3.1 Interaction Overview

1. The interface is run on a laptop, thus the screen acts as the performance space.

2. The eye-tracker is placed on a table in front of a user within its optimal range of operation (around 30-35 cm away)

3. The user calibrates the eye tracker with their sitting position before each use, and looks at the software interface (see Fig. 2), where a small green square indicates the current position of the user's gaze.

4. The user then plays music by looking at the various musical keys (GUI buttons). Additional controls for playing various types of diatonic chords and varying musical dynamics are also provided as buttons, which are similarly activated through gaze.

3.2 Physical Model of a Xylophone

The xylophone comprises multiple wooden bars each corresponding to a different musical note, which are struck using a mallet to generate sound. On breaking down the challenge of physically modelling a xylophone, it is observed that a wooden bar is the basic component of a xylophone. Wooden bars can be physically modelled based on the equations of the ideal bar provided by Bilbao in [15]. Chaigne and Doutaut also present the physical description of a xylophone bar in [16]. On inspecting the structure of a xylophone closely, it can be ascertained that the mallet is the 'exciter', and the wooden bars are 'resonators' in a

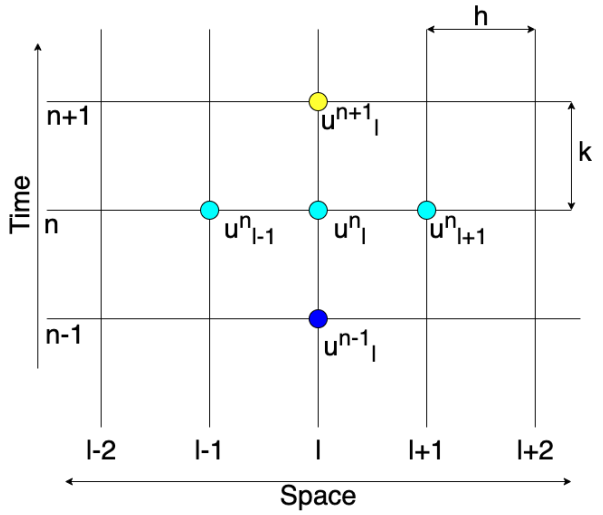


Figure 1. Graphical Representation of the Finite Difference Approach Used to Physically Model the Xylophone

xylophone. Vibrations are created when a mallet strikes a bar, which are sustained by the bar to generate an after-sound containing only the fundamental frequency of the bar [16] [17].

For our purposes, the sound of a xylophone bar was simulated by physically modelling an ideal bar using finite difference methods. To account for the decay of sound, frequency based and non-frequency based damping terms were incorporated in the equation of an ideal bar given in [15]. Further, the excitation action was simulated by using a raised cosine (Hanning window) as an initial excitation for the model. It was found that this method generated results that could be tuned to sound close to how an actual xylophone sounds.

3.2.1 Mathematical Equations

Bilbao provides in [15] the definition of the state of a system distributed in space as:

$$u = u(x,t) \tag{1}$$

The state equation of an ideal bar contains the second-order partial derivative of the state of the system in space and the fourth-order partial derivative of the state of the system in time, related by the stiffness coefficient of the ideal bar. It is given by Bilbao in [15] as:

$$u_{tt} = -\kappa^2 u_{xxxx} \tag{2}$$

Where u_{tt} denotes the second-order spatial derivative of u in time, u_{xxxx} denotes the fourth-order spatial derivative of u in space and κ denotes the stiffness coefficient of the bar.

The stiffness coefficient of the bar is calculated as:

$$\kappa = \sqrt{EI/\rho A} \tag{3}$$

Where,

- *E* denotes the Young's modulus of the material of the bar
- *I* denotes the moment of inertia of the bar
- *A* denotes the cross-sectional area of the bar, and
- ρ denotes the material density of the bar

The cross-sectional area of the bar is calculated by the product of its width, *b* and height, *x*. The following equation, obtained from [16] is used to calculate the moment of inertia of the ideal bar using the same two quantities:

$$I = bx^3/12 \tag{4}$$

However, equation (1) only describes an non damped, ideal bar. In real life applications, like the physical model of our xylophone, the bars are not be completely ideal and they contain damping factors that determine how quickly the generated sound fades away. The equation for a damped ideal bar contains two extra terms, one for frequency based damping and another for non-frequency based damping.

The non-frequency based damping term is given as:

$$-2\sigma_0 u_t, \tag{5}$$

The frequency based damping term is given as:

$$2\sigma_1 u_{txx} \tag{6}$$

Where σ_0 and σ_1 are the non-frequency based and frequency based damping coefficients, respectively.

Combined, equations (2), (5) and (6) give us the final equation that was used to model the bars of our xylophone:

$$u_{tt} = -\kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx} \tag{7}$$

Equation (7) contains the following partial derivatives in addition to the ones in equation (2):

- u_t, which is the first order partial derivative of the state of the system in terms of time
- u_{txx}, which is the third order mixed partial derivative of the state of the system, in terms of both time and space

All the partial derivatives need to estimated using finite differencing methods in order to obtain the final equation for the state of the system. Furthermore, the stability of a finite differencing scheme is determined using a stability condition, that incorporates the spatial step of the discretised grid, *h* and also its temporal step, *k*. The temporal step, *k*, is calculated as the inverse of the sampling frequency.

The stability condition for a damped ideal bar, described by equation (7) is given as:

$$h = \sqrt{(4k(\sigma_1 + \sqrt{\sigma_1^2 + \kappa^2}))/2} \tag{8}$$

Finally, we arrive at the fully discretised state-update equation for an ideal bar, which is obtained by the expansion of all partial derivatives in equation (7) using finite differencing methods described in [15]. It is obtained as:

$u_l^{n+1} = 2u_l^n - u_l^{n-1} - \kappa^2 k^2/h^4(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n) - 2\sigma_0 k(u_l^n - u_l^{n-1}) + \sigma_1 k/h^2(u_{l+1}^n - u_{l+1}^{n-1} - 2u_l^n + 2u_l^{n-1} + u_{l-1}^n - u_{l-1}^{n-1})$

Here, u_l^n refers to the output of the system at the current time state (n) and at the current spatial stage (l). In this way, the locations of all terms in equation 4.9 can be ascertained.

3.3 GUI Elements and Layout

Twelve musical keys were provided, corresponding to an octave in an equitempered scale. Similarly to previous work [7], the musical keys of the xylophone were placed in a circular layout, with an inactive region in the center. The musical keys were coloured yellow, changing to green when struck to provide visual feedback. A green on-screen pointer was also incorporated to indicate instantaneous gaze position.

Finally, it was decided to not place GUI elements towards the extreme edges of the interface. This was done keeping in mind the fact that the accuracy of eye-trackers is higher towards the center of the screen, and it reduces closer to the edges of the screen.

3.4 Tuning the Physical Model

Once a working iteration of the physical model was obtained, it had to be tuned to sound like a xylophone. As a part of this process, the pitch and timbre of each key had to be tuned. The damping terms present in the state equation of an ideal bar were used to tune the sustain of the output. On the other hand, the length and height of the bar were altered to change the pitch of the model [17].

A collection of anechoic recordings of xylophone notes was used as a reference to tune the damping of the physical model by ear. These are made available for free by the University of Iowa [1]. The damping factors were altered until the output of the model seemed to have the same sustain as the anechoic recordings.

To arrive at the correct pitch for the xylophone bars of the ADMI, the MATLAB FFT function was used to analyse the output of multiple versions of the ideal bar. The next step was to tune each key to a fixed fundamental frequency. The fundamental frequencies of the semitones in the octave C4 - B4 are widely known, therefore the task at hand was to match the frequency of each of the 12 keys to that of the correct semitone. After some initial attempts, it was ascertained that increments in the height and the length of the ideal bar had the effect of increasing its fundamental frequency. It was noticed that an increase in the height of a virtual xylophone bar reduced the spatial resolution of the finite difference scheme, whereas an increase in the length had the opposite effect. Therefore, it was decided to only alter the length of the bar to bring about changes in the fundamental frequency. Bar lengths corresponding to one musical octave were thus ascertained, and can be seen in Table 1.

[1] https://theremin.music.uiowa.edu/MIS-Pitches-2012/MISxylophone2012.html

Note	Length (in metres)
C4	0.6810
C♯4	0.6720
D4	0.6520
D♯4	0.6460
E4	0.6270
F4	0.6191
F♯4	0.6030
G4	0.5690
G♯4	0.5356
A4	0.5270
A♯4	0.5120
B4	0.5061

Table 1. This table shows the different notes selected for the interface, and the length (in metres) of the virtual wooden bar for each fundamental frequency

3.5 Gaze Tracking Functionality

1. The Tobii X2-30 was interfaced with the computer using the Tobii EyeTracker Manager software on the computer [2].

2. The Tobii SDK [3] was used to interface the eye tracker with JUCE, which obtained instantaneous gaze data from the eye tracker at a rate of 40 Hz. The raw gaze tracking data was obtained separately for the left and right eyes.

3. A data description process was undertaken to permit fluid sonic interaction with the physical model. First, the two gaze points received at every instance were averaged to find a central gaze coordinate. Next, a gaze-detection algorithm was implemented to interpolate the data and allow smooth gaze based control. The gaze detection algorithm used was presented by Kumar et al. in [18], and its pseudo-code given by Kumar in [19]. This algorithm takes into account the previous coordinate at every step, and then compares it to a predefined 'saccade threshold' which is the radial distance from a point beyond which the subsequent coordinate is classified as a saccade. It then stores coordinates not described as saccades in a 20 sample window and computes the weighted mean of this window to obtain a final coordinate location. The on screen gaze pointer was programmed to follow this interpolated gaze coordinate location.

   The formula used to calculate weighted mean in the saccade detection algorithm is given in [19] as follows:

   p_{mean} = (1·p_0 + 2·$p1$ + ... + n·p_{n-1}) / (1 + 2 + ... + n)

[2] https://www.tobiipro.com/product-listing/eye-tracker-manager/

[3] https://www.tobiipro.com/product-listing/tobii-pro-sdk/#Downloads

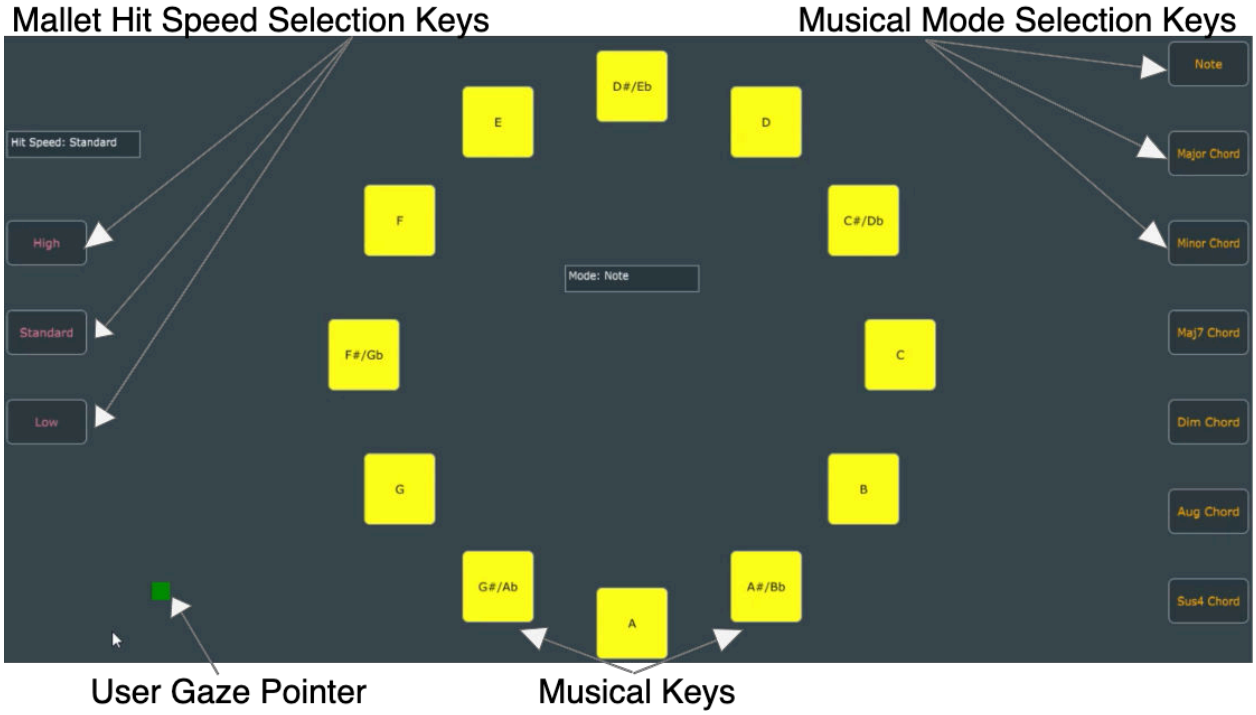


Figure 2. User Interface of the Final Iteration of the Virtual Xylophone

4. Finally, a function was implemented to trigger any button that contained the on screen gaze pointer.

Audiovisual demonstrations of the xylophone and ADMI can be accessed on Dropbox [4]. The source code of the project is available on GitHub [5].

4. EVALUATION

We carried out a brief user study to assess the prototype in terms of

- **Usability** (captured using the *System Usability Scale (SUS)* [21])
- **Desirability and Playability** (using word selection techniques proposed in [22])

4.1 Participants

A convenience sample of nine participants (students at Aalborg University Copenhagen) was used for the evaluation. They had different levels of music training and general experience using eye-trackers. Each participant was informed about the purpose and duration of the experiment, and relevant consent was obtained from each participant. No sensitive personal information was collected about the participants and they had the option to withdraw at any time.

[4] https://www.dropbox.com/sh/rihsgakhpag0b6i/AACqLc5Eu-oiCa4pE-fXCWUIa?dl=0

[5] https://github.com/mrkandpal/EyeTrackingJUCE

4.2 Experimental Setup

The experiment was carried out at the Multisensory Experience Lab at Aalborg University. It was hosted on an Asus laptop running the JUCE application with integrated eye tracking and also provided experiment instructions. Furthermore, a demo video of the interface was also shown on this laptop, which documented the tasks each participant had to perform as a part of the experiment.

4.3 Procedure

The following steps were followed by each participant:

1. First, the participant was briefed about how eye tracking works and the purpose of building this ADMI. The eye tracker was then calibrated.
2. The participant was then given some time to get accustomed to the ADMI and its interaction mechanism. Once a participant became comfortable with the ADMI, they were asked to play the C major scale on it.
3. The participant was then presented with a set of 3 different tasks, each of which was described to the participant through a video tutorial.
4. Finally, the participant was presented with the SUS survey and the desirability assessment word-list.

4.4 Data Analysis

The data were collected and analyzed in MS Excel. The key outcomes were as follows:

Sr. No.	Statement	Valence	Aggregated Score
1	I think that I would like to use this system frequently.	+	2.34 ± 0.82
2	I found the system unnecessarily complex.	-	1.67 ± 0.67
3	I thought the system was easy to use.	+	3.12 ± 0.74
4	I think that I would need the support of a technical person to use this system.	-	1.67 ± 1.05
5	I found the various functions in the system to be well integrated.	+	4.00 ± 0.67
6	I thought there was too much inconsistency in the system.	-	2.34 ± 1.05
7	I would imagine that most people would learn to use this system very quickly.	+	3.78 ± 1.13
8	I found the system very cumbersome to use.	-	3.45 ± 0.95
9	I felt very confident using the system.	+	3.34 ± 0.94
10	I needed to learn a lot of things before I could get going with this system.	-	1.45 ± 0.50
11	**Overall SUS Score**		**65.5 ± 11.53**

Table 2. Results from the SUS questionnaire. Each statement was rated on a discrete scale from 1-5, where 5 indicated complete agreement, and 1 was complete disagreement. As is evident from the table, the statements alternated in terms of their valence (+/-). The aggregated values are presented as **mean** $\pm$ **standard deviation**. An SUS score of 68 is considered average [20]

1. SUS Score and its subcomponents

2. The occurrence frequencies of the selected words in the desirability assessment

5. RESULTS

Overall, we observed that the participants were able to use the interface and successfully perform the tasks. Table 3.5 summarizes the item-wise and overall aggregated SUS scores from the usability evaluation. The obtained mean **(65.5)** was slightly lower than the documented average from research [20]. The ratings indicate that that participants found the various functions of the interface to be well integrated **(Item 5)** and the overall system easy to use **(Items 2, 3, 4, 7, 10)**. Despite this, users appear to have found the system cumbersome **(Item 8)** and they did not express wishing to use it frequently **(Item 1)**.

Table 3 summarizes the outcomes of the desirability assessment. The majority of frequently- and somewhat used adjectives are positive in valence, with the exception of words such as *frustrating, clumsy, slow* and so on as shown in the table. The combined results are discussed in detail in the next section.

6. GENERAL DISCUSSION

In this work, we designed and developed a JUCE-based technical framework to explore the feasibility and interactive potential of gaze-controlled physical models for musical expression. We were successful in realizing a stable working prototype with simple note and chord generation capabilities, and our usability test showed that the graphical interface was operable without excessive difficulty.

Looking closely at the SUS results, it is clear that the participants did not find the system to be prohibitively complex or confusing. This could be attributed to the simplicity of the interface and its available interactions. The straightforward layout with clearly separated sections pertaining to hit velocity, note/chord mode and ring-shaped note triggers may have contributed to the participants' ease in operating the interface. The ring-shaped design was based on past approaches [7, 13] and appears to be a suitable layout shape for note triggers in such applications. However, it is equally likely that the simplicity of the interface contributed to participants not finding it engaging enough to wish to use it frequently. It will be interesting to see how participants' impressions of ease-of-use scale when the system itself scales in complexity and functionality. It is likely that the addition of new expressive affordances will lead to a similar steepening in learning curve to that seen in [7].

In spite of the stated ease-of-use, the overall usability of the system was rated below average, partially because participants tended to find the system cumbersome to use. We strongly suspect that this was in large part due to spatiotemporal inaccuracies in the eye tracking as well as calibration drift, both of which are well-documented issues with present eye tracking technology [5] and past ADMIs [7, 9]. Whilst it is likely that newer hardware and firmware will gradually resolve these issues, a more sophisticated digital filtering algorithm (e.g, Kalman filter [23]) could have improved the usability of our system, and will be explored in future versions of the framework. Another possible cause of the user difficulty may have been the relatively brief training phase; it is likely that participants would have found the system easier to use with more practice, as also highlighted in [7].

The findings from the desirability evaluation aligned well with the SUS results, in that participants highlighted themes related to simplicity and intuitiveness, but also frustration and clumsiness. Whilst the overall positive disposition of participants can be interpreted in the system's favour, it is likely that being in the same room as the experimenter during the test may have introduced a social desirability bias into the responses of the participants. Future studies should adopt blinded designs with anonymous responses when conducting these assessments so as to minimize this

Frequently Used			**Somewhat Used**			**Seldom Used**		
Word	Valence	#Occ	*Word*	Valence	#Occ	*Word*	Valence	#Occ
Accessible	+	8	*Clumsy*	-	3	*Annoying*	-	2
Entertaining	+	5	*Easy-to-Use*	+	4	*Appealing*	+	2
Frustrating	-	6	*Responsive*	+	4	*Satisfying*	+	2
Intuitive	+	5	*Slow*	-	4	*Unpredictable*	-	2
Simplistic	+	6	*Stimulating*	+	3			
Straightforward	+	7	*Uncontrollable*	-	3			
			Unconventional	+/-	3			
			Usable	+	3			

Table 3. Results from the desirability assessment. The words are segregated based on how many participants selected them (out of a total nine) into *frequently used, somewhat used, and seldom used*. The interpreted valence of each word as well as its number of occurrences is also indicated. Words selected by less than two participants are omitted here

form of bias. However, the main takeaway from both assessments appears to be that the participants found the interface straightforward and easy-to-follow, but that its limitations in terms of tracking accuracy and expressive potential negatively contributed to their overall perceptions of its usability.

This work had several limitations, both in terms of design and evaluation. The interaction design provided only limited parametric control over the physical model properties. Future versions should enable a greater degree of user control over expressive parameters related to articulation, strike position, timbre, and dynamics, aside from adding a wider selection of instruments and possibly step sequencing functionality like in past DMIs [7, 8]. The parameter mappings should explore the use of embodied conceptual metaphors as a means to create meaningful movement-sound links [24]. In terms of technical considerations, support should be added for several eye-tracker models, and rigorous assessments of computational load and tracking accuracy should be conducted to ascertain the pros and cons of the various eye-tracker morphologies [5]. Our evaluation assessed usability and desirability with convenience-sampled participants, but did not investigate user perceptions of the expressive capabilities of the system with the user group. An evaluation procedure similar to that done in [7] is a good reference for future versions of our framework. Lastly, an interdisciplinary approach that involves end-users (e.g, ALS patients) in the design process would be the ideal approach to ensure that the developed system services its target audience.

7. CONCLUSIONS

In this work, we were able to establish a feasible digital musical interface that integrated gaze tracking with real-time physical modelling synthesis in a scalable, open-source prototype. Whilst the interface was found to be a feasible means for gaze-driven music performance our qualitative results indicate that the utility of the interface can be enhanced by expanding the possibilities for expressive control over the physical model There is indeed much room for further development, improvement, and evaluation, but we believe that the present work provides a robust technical framework for the future exploration of gaze-controlled physical models for musical expression. We hope that this can serve as a step towards enabling individuals with severe physical impairments to experience a rich world of digitally augmented creative musical experiences.

Acknowledgments

DK was responsible for system design, development and evaluation. PK assisted with technical implementation and SS supervised the project. All authors collaborated on writing the manuscript, and approved of the final version. We would like to thank the usability test participants for their time and inputs. This work is partially funded by NordForsk's Nordic University Hub, Nordic Sound and Music Computing Network NordicSMC, project number 86892. The participation to the conference was supported by the European Art Science and Technology network (EASTN-DC).

8. REFERENCES

[1] J. Shepherd and P. Wicke, *Music and Cultural Theory*. Polity Press ; Published in the USA by Blackwell, 1997.

[2] J. Malloch, D. Birnbaum, E. Sinyor, and M. M. Wanderley, "Towards a new conceptual framework for digital musical instruments," in *Proceedings of the 9th international conference on digital audio effects*. Citeseer, 2006, pp. 49–52.

[3] F. Grond, K. Shikako-Thomas, and E. Lewis, "Adaptive musical instruments (amis): Past, present, and future research directions," *Canadian Journal of Disability Studies*, vol. 9, no. 1, pp. 122–142, 2020.

[4] E. Frid, "Accessible digital musical instruments—a review of musical interfaces in inclusive music practice," *Multimodal Technologies and Interaction*, vol. 3, no. 3, p. 57, 2019.

[5] P. Majaranta and A. Bulling, "Eye tracking and eye-based human–computer interaction," in *Advances in physiological computing*. Springer, 2014, pp. 39–65.

[6] L. C. Wijesekera and P. N. Leigh, "Amyotrophic lateral sclerosis," *Orphanet journal of rare diseases*, vol. 4, no. 1, pp. 1–22, 2009.

[7] R. Ramirez and Z. Vamvakousis, "The eyeharp: A gaze-controlled digital musical instrument," *Frontiers in Psychology*, vol. 7, 2016.

[8] W. Payne, A. Paradiso, and S. K. Kane, "Cyclops: Designing an eye-controlled instrument for accessibility and flexible use," 2020.

[9] S. Greenhill and C. Travers, "Focal: An eye-tracking musical expression controller." in *NIME*, 2016, pp. 230–235.

[10] T. Hermann, A. Hunt, and J. G. Neuhoff, *The Sonification Handbook*. Logos Verlag Berlin, 2011.

[11] C. Abras, D. Maloney-Krichmar, J. Preece *et al.*, "User-centered design," *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, vol. 37, no. 4, pp. 445–456, 2004.

[12] N. Davanzo and F. Avanzini, "Hands-free accessible digital musical instruments: Conceptual framework, challenges, and perspectives," *IEEE Access*, vol. 8, pp. 163 975–163 995, 2020.

[13] S. Valencia, D. Lamb, S. Williams, H. S. Kulkarni, A. Paradiso, and M. Ringel Morris, "Dueto: Accessible, gaze-operated musical expression," in *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, 2019, pp. 513–515.

[14] J. O. Smith III, "Viewpoints on the history of digital synthesis," in *Proceedings of the International Computer Music Conference*. International Computer Music Association, 1991, pp. 1–1.

[15] S. Bilbao, *Numerical sound synthesis: finite difference schemes and simulation in musical acoustics*. John Wiley & Sons, 2009.

[16] A. Chaigne and V. Doutaut, "Numerical simulations of xylophones. i. time-domain modeling of the vibrating bars," *The Journal of the Acoustical Society of America*, vol. 101, no. 1, pp. 539–557, 1997.

[17] J. L. Moore, "Acoustics of bar percussion instruments," Ph.D. dissertation, The Ohio State University, 1971.

[18] M. Kumar, J. Klingner, R. Puranik, T. Winograd, and A. Paepcke, "Improving the accuracy of gaze input for interaction," in *Proceedings of the 2008 symposium on Eye tracking research & applications*, 2008, pp. 65–68.

[19] M. Kumar, "Guide saccade detection and smoothing algorithm," *Technical Rep. Stanford CSTR*, vol. 3, p. 2007, 2007.

[20] J. R. Lewis, "The system usability scale: past, present, and future," *International Journal of Human–Computer Interaction*, vol. 34, no. 7, pp. 577–590, 2018.

[21] J. Brooke, "Sus: a "quick and dirty'usability," *Usability evaluation in industry*, vol. 189, 1996.

[22] J. Benedek and T. Miner, "Measuring desirability: New methods for evaluating desirability in a usability lab setting," *Proceedings of Usability Professionals Association*, vol. 2003, no. 8-12, p. 57, 2002.

[23] M. Toivanen, "An advanced kalman filter for gaze tracking signal," *Biomedical Signal Processing and Control*, vol. 25, pp. 150–158, 2016.

[24] S. Roddy and B. Bridges, "Addressing the Mapping Problem in Sonic Information Design through Embodied Image Schemata, Conceptual Metaphors, and Conceptual Blending," *Journal of Sonic Studies*, no. 17, 2018.

THE SINGING SHOWER: A MELODY-SENSITIVE INTERFACE FOR PHYSICAL INTERACTION AND EFFICIENT ENERGY CONSUMPTION

Yann Seznec
KTH Royal Institute of Technology
yannse@kth.se

Sandra Pauletto
KTH Royal Institute of Technology
pauletto@kth.se

ABSTRACT

This paper proposes an interface based on melodic input, encouraging a user to sing in order to interact with a device. We describe the early stages of designing and prototyping a sound-reactive shower, which is controlled by a user singing to control the flow of water. We then discuss the implications of this design with regards to energy and resource efficiency, as well as being a form of provocation and experimental interface design. Interaction design has an important role to play in facilitating sustainable behaviour in the household. We propose that sonic interfaces such as this can contribute to this area of work, and that an interface based on melodic input can be used to seamlessly activate and deactivate a system while using hands and vision to accomplish other actions and reducing energy consumption. In this paper, the prototype is described, evaluated and results are discussed. Finally, directions for future work and extensions of this system are proposed.

1. INTRODUCTION

The resources used in the home environment are, for the most part, invisible [1]. Whilst in the past the energy for a household would have been generated inside the home with fire, coal, or peat, in modern times this is no longer seen, heard, or felt. The electricity to power our appliances and the gas burned to create our heating and hot water is in many ways now an abstract concept. Although many would like to lower their resource use, it can be hard to quantify or even imagine what it means [2]. The study described in this paper is part of a larger project, Sound for Energy [1] , which proposes that sonic interaction design can offer one method for addressing this issue.

This paper describes the design of the *Singing Shower* - a system enabling a user to control the flow of water in a shower with their singing. We will describe the development of the first prototype as well as our initial user tests. Results are then discussed and directions for future research are suggested.

The Singing Shower is in some ways a experimental interface for engaging with home energy use and water consumption through sound. However we will argue that the prototype, even in its current early stage, manages to address and challenge fundamental issues in interface design and home resource use, through the medium of playful sonic interaction.

[1] https://soundforenergy.net

Copyright: © 2022 Yann Seznec et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. BACKGROUND

The design of how we interact with everyday objects can strongly contribute to promoting a more sustainable behaviour [3]. This is increasingly true as homes begin to be equipped with smart meters and IoT based Home Energy Management Systems (HEMS), which can provide information, awareness, and control possibilities to end-users [4]. However, research has shown that existing visual displays of energy costs and savings are not effective [5,6]. Studies have found that current displays are designed for an unrealistic "resource man" interested and able to micromanage energy resources, rather than a real user living in a "domestic mess" [7]. Instead, users need to be inspired and have fun in order to be engaged [8].

While several studies (e.g. [9–11] have attempted to overcome issues of engagement and design, only rarely has non-speech sound been considered as primary mode of communication and interaction (e.g. [12–14], and only recently has this area emerged as a topic in its own right within the sonic interaction design and sonification fields [15, 16]. Research has shown that everyday sounds such as knocks, footsteps or vocal sounds, are extremely informative despite being perceived as mundane, with the advantage of requiring little conscious attention or effort (e.g. [17–19]. Additionally, researchers have identified an unpopulated design space in the domestic soundscape for novel sonic interactions (e.g. [20]. These novel sonic interactions can address issues related to sustainable behaviour by harnessing existing domestic sound behaviours and attitudes (e.g. singing happily under the shower), while at the same time increase awareness, control and playfulness. Developing sonic interactions that are familiar, personal and even loved by their users could facilitate both long term behavioural change and a feeling of satisfaction.

In this paper, we present a prototype of a sound-reactive shower, which is operated by a user singing to control the flow of water. This sonic interaction makes use of the personal and playful experience of taking a shower, while at the same time promoting an efficient way to control water consumption.

The starting point is the concept of a so-called *Navy Shower*. This is a technique developed in military envi-

ronments, particularly on naval ships, in order to conserve water and heating to maximise efficiency in the use of resources. The main strategy of a Navy Shower is to turn the flow of water on only when necessary for rinsing the body, and turn off the water while applying soap [21]. This can have significant impact on resource use, with at least one study claiming a 95% reduction in average water usage, leading to a potential saving of 15,000 gallons per year per person [22]. Domestic Hot Water (DHW) usage has also been identified as a key area of high energy use, remaining relatively consistent in recent decades whilst other forms of heating have been made more efficient [23]. The Navy Shower approach offers a proven method for addressing this area of energy inefficiency in the home.

The shower offers an opportunity for novel sonic interactions through the medium of singing. Singing in the shower is a well documented phenomenon, with references dating back to at least the 14th century [24] and certainly occupying a clear place in popular culture. A number of potential reasons have been proposed for its popularity, often referencing the unusually sonic characteristics in bathrooms due to size, tiling, and generally more reflective surfaces than elsewhere in the home.

Additionally, singing in the shower is clearly a playful experience. Playfulness and ludic experiences have a unique role in human lives [25]. We play to learn new things, create new memories, and develop new values and understandings. Recent research is starting to investigate how we can design new interactions harnessing the power of playfulness [26–28]. In line with this approach, in this project we aim to leverage hedonic and playful opportunities to encourage greater resource efficiency.

Sound, and in particular the human voice, can function as a powerful tool for computer interaction and physical interface control. Voice control systems have been under development since at least the early 1980s (e.g. the VOTAN V5000 system [29]). Voice interaction has more recently become essentially "domesticated" within modern homes [30–32], with products such as the Amazon Alexa or Apple Homepod. However this domestication comes at a cost, as the seamless experience offered by modern voice control is both physically and ethically encumbered [33] as it relies on massive servers, raises significant privacy concerns, and creates issues related to bias (in regards to accents, vocabulary, and fluency, etc) [34–37]. In this project we are thus attempting to retain the advantages of voice control (e.g. freeing hands and eyes), whilst avoiding its problematic aspects.

The device presented here regulates the flow of water in the shower in response to a user singing. This sonic interaction method enables efficient and touch-free control of water, simplifying (and possibly augmenting) the Navy Shower approach to efficient resource control.

The result is a playful, experimental, and in some ways provocative device that builds on prototyping approaches that engage with the "messy, intimate, and contested aspects of everyday life" [38, p. 2549].

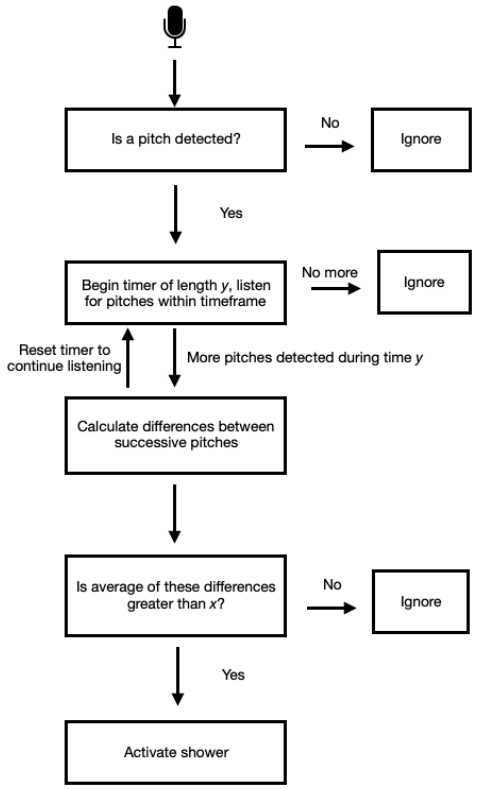


Figure 1. Flow chart indicating the structure of the machine listening code

3. SYSTEM DESIGN

In order to address these challenges we have chosen to develop a melody-reactive binary control system which detects only whether or not a user is singing. This deceptively simple concept opens a number of questions in terms of technical design, interaction design, and social and cultural considerations. Limiting the input to melody and the output to a binary state allows the system to completely avoid any recording, bypassing any potential privacy issues. The system can be entirely self-contained (and un-encumbered), whilst retaining the major advantages of voice control in terms of hands-free usage and seamless touch-free interaction.

In this design, the binary control is dependant on continuous singing - the water flows only whilst the singing is occurring. This is an important distinction from clap detectors, for example, which function as latching switches. This is a deliberate design choice that aims to reinforce the physical connection between the user and the water flow, with their voice providing the metaphorical "power" required to shower. This is designed to provide an embodied experience of energy consumption [39, 40], and to make visible (and audible) an otherwise invisible process of intensive resource use [2].

3.1 Melody Detection Software

For our first prototypes we built a system in Pure Data (Pd) [2] running on a Bela microcomputer. The code can be found on the author's GitHub page. [3]

We developed a machine listening system that reacts mainly to a series of different recognized musical pitches within a short timeframe. This approach biases the system towards short melodic phrases with relatively large differences in pitch. This was deemed acceptable for this prototype as it feels playful and reacts quickly - in addition there are a number of variables (e.g. timeframe length, size of differences in pitch) that can be adjusted during testing.

The prototype contains an electret microphone connected directly to the Bela Mini microcontroller (using a resistor attached to 3.3v power [4]). The Pure Data code running on the Bela is using the *sigmund~* object to listen to the incoming audio. If a pitched note is detected on the audio, this pitch will be stored in an array and a timer will be activated. This provides the first level of filtering on the audio, as the pitch will only be detected above a certain threshold of recognizability (set using the *threshold* variable on the *sigmund~* object), therefore a fair amount of non-musical sonic activity will be ignored immediately.

Following this, any further notes that are detected within the timeframe (variable *y* in Fig. 1) dictated by the timer will also be stored in an array. The difference in MIDI note value between each successive note is calculated, as is the average of these differences. If this average is above a certain value (variable *x* in Fig. 1), and the timer is still activated (meaning that pitched notes are still being detected), the digital output connected to the shower solenoid is set to 1 and the shower turns on. If not, the output value is set to 0 and the array is cleared.

3.2 Hardware

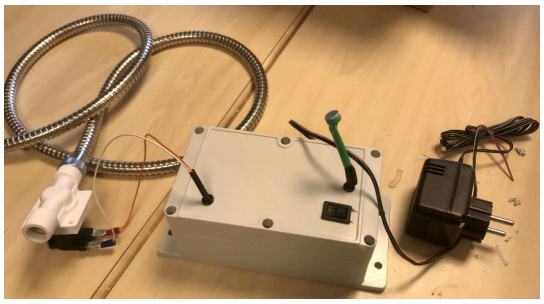


Figure 2. Hardware prototype

On the Bela, the output from Pd is sent to a digital pin, which is connected to a relay. This relay controls the state of a water valve solenoid, which is placed in between the hot water mixer and the shower head of a standard shower.

The majority of the electronics are housed in a waterproof enclosure in order to protect it from moisture, apart from the electret microphone which protrudes from the box. A large rechargeable battery powers the Bela, however this was not sufficient to power the solenoid, and no satisfactory method for integrating the water valve into the enclosure was found. Therefore the water solenoid was hung outside of the enclosure for testing, and a standard 12V wall power supply was used to power the relay for this initial prototype. A short demo of the singing shower can be viewed online. [5]

4. EVALUATION METHODS

A pilot user experience test was devised in order to evaluate how users would react to a melody-sensitive interface, and to test the robustness of software and hardware. The prototype was installed in a shower in the department bathroom at KTH in Stockholm. Eight participants (M=3; F=5; average age: 31) were recruited from within the department and consisted mainly of students and staff. They were given a brief description of the prototype and asked to stand near the shower and sing to control the flow of water for as long as they felt comfortable. For this initial test, the users were not asked to take a shower. The participants were then left alone to complete the task. Audio recordings were made of their tests, and following that they were asked to complete a User Experience Questionnaire (UEQ) [6] [41] and answer in writing a few open ended questions about the experience.

Prior to this testing, the prototype was installed in the lead author's shower for several days, in order to evaluate the hardware and software and to create video documentation. The author and one other member of the household used the prototype to successfully take (singing) showers.

5. RESULTS

5.1 System Functionality

The eight participants interacted with the system for an average of 3.5 minutes each (min duration 1 minute; max duration 7 minutes). On average the water flow was turned on 18 times, of which 13 were deliberate. On average the flow was stopped 13.5 times, of which 10.9 were deliberate. Thus the success rates of the machine listening and mechanical control system were 72% for switching on and 81% for switching off.

An analysis of the recordings reveals that participants tested the system in a number of ways other than singing a tune. They tried to whistle, speak, clap, fool the system by playing music from a mobile phone, and test single pitches by saying a single pitched vowel - in addition one trained classical singer tried a number of modern extended singing techniques. Clapping, single pitched sounds vowels, and music did not turn on the system. Speech sometimes activated the shower, whistling seemed to confuse the system, and the modern extended singing techniques had mixed results.

[2] https://puredata.info

[3] https://github.com/yannseznec/singing-shower-software

[4] https://forum.bela.io/d/62-cheap-electret-microphone-how-to

[5] https://www.youtube.com/watch?v=8UUXhprBt7U

[6] https://www.ueq-online.org

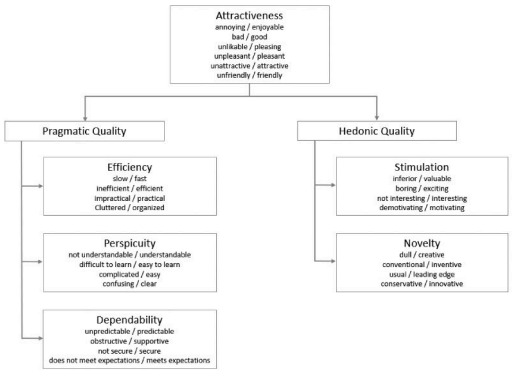


Figure 3. UEQ scales: image from the UEQ Handbook in `https://www.ueq-online.org`

5.2 User Experience

The User Experience Questionnaire provides validated scales to measure six user experience factors which describe both Pragmatic and Hedonic qualities of an experience: Attractiveness (Do the users like the experience?); Perspicuity (Is it easy to learn what to do, and get familiar with?); Efficiency (Can the task be achieved without unnecessary effort?); Dependability (Does the user feel in control of the interaction?); Stimulation (Is it exciting and motivating?); Novelty (Is the experience innovative, creative, interesting?). In total 26 scales are used: 4 scales contribute to each factor with the exception of Attractiveness which uses 6 scales. A summary of the scales can be seen in Fig. 3.

The main advantage to using such a tool for quantitatively measuring the main aspects of user experience is that it provides a simple but robust way to compare results between this initial version and future versions of the system. Additionally results are automatically set in relation to existing values from a benchmark data set containing data from 21175 persons from 468 studies concerning different systems and products.

While results from this pilot test should be taken carefully given the low number of participants, the UEQ questionnaire can provide an initial snapshot of the main trends in the user experience of the system.

From Fig. 4, we can see that Attractiveness and Hedonic qualities are rated positively, while the Pragmatic quality is considered less positively. If we look at the six contributing factors: Novelty and Stimulation are considered very positively, while Dependability and Efficiency score quite low. Perspicuity does not score sufficiently high, however it seems to be less problematic. When comparing against the benchmark provided by the authors of the UEQ, Attractiveness scores just below average, Stimulation and Novelty are above average and excellent respectively, while Perspicuity; Efficiency; Dependability are all in the range of the 25% worst results. It seems clear that, while the Hedonic qualities could get through even when using a very initial prototype, Pragmatic Qualities need further iterations of the system to be perceived to be functioning always as expected.

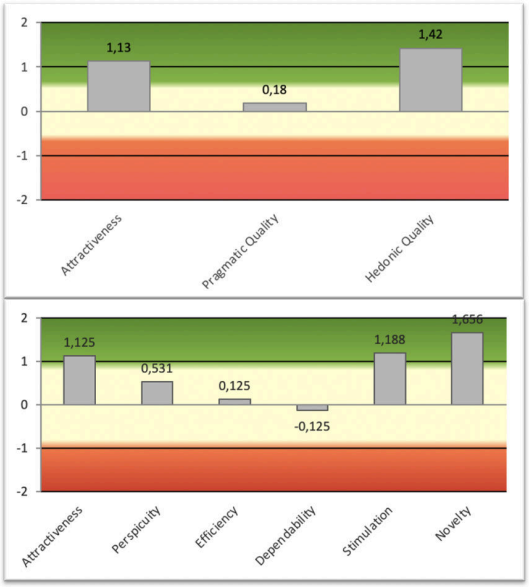


Figure 4. Summary of results: Top Graph - Attractiveness, Pragmatic Quality and Hedonic Quality; Bottom Graph - Attractiveness; Perspicuity; Efficiency; Dependability; Stimulation; Novelty

When looking at the results for the individual items within the factors (Fig. 5), we find that the system scored very high for friendliness, inventiveness and innovation, then follows interest and excitement; while the system scores negatively for predictability, support of the task, and security (i.e. it is not always sure that it would work as expected).

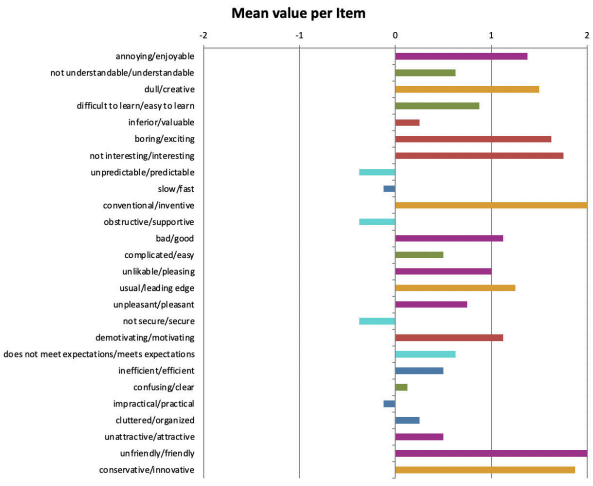


Figure 5. Results by Items

5.3 Open Ended Questions

After the test, we asked the participants to describe their experience, to describe how they thought the singing controlled the flow of water, and whether they would feel comfortable to use such a system at home. Finally they were

free to add any further comment.

The experience was described as interesting, intuitive, joyful and energising. One person commented that the system worked as expected most of the times. With regard to control participants commented that it was pretty smooth, and easy. They noticed some delay between the start of singing and the water flowing, additionally they thought that a minimum sound level was required for the singing to be recognised as quieter singing seemed to be missed at times. One participant commented: "I tried to make some different kinds of singing to see if it affected the shower in some way, but did not really understand if it did, or if it just reacted to sound and was a bit buggy." In terms of using this system at home participants noted that they would not have a problem, and might prefer it over a regular shower and that it was fun, however final usage might depend on the mood of the day. One person noted: "I don't think that I would like to have it in my house. It would be quite exhausting to sing all the time you want the water to run." One participant expressed concern about having the system connected to the internet, recording sounds (they were not aware that the system does not need internet connection or recording sound).

Finally, we asked one person, other than the first author, to take a normal shower with our system. We report here the feedback which is promising and appears to align with the design goals:

> The way the water responded to my voice gave me the sense that I was physically powering the shower with my own breath. For the first few seconds I had to experiment a little with the volume at which I was singing, but once I got the hang of it, it felt easy. I was surprised by how little I needed the water to be on to clean myself effectively. But I was also surprised by how much I enjoyed the experience and how empowering and fun it was to sing to control the shower.

6. DISCUSSION

Despite being an initial prototype, the system demonstrated great potential as a playful interface. This was reflected both in the UEQ results for the Hedonic Qualities and Attractiveness, and the user feedback and audio recordings made during testing. It is also reassuring that Perspicuity - which refers to learnability and ease of use - despite not yet scoring very high, does not seem to be the major culprit in the relatively low score of the Pragmatic Quality. As the technical robustness is improved, we speculate that learnability and ease of use will consequently improve. Overall, these results reflect the cohesive design framework, and in particular the clear reference to the existing cultural phenomenon of singing in a shower seems to be appealing. The test gave us a broad sense of how users would enjoy the system, test its limitations, and how comfortable they were with the idea of controlling the flow of water with their voices.

The technical functionality of the prototype was more than sufficient for this initial test, with significantly more successful interactions than failures. The spectrograms (Fig. 6) from the audio recordings of the test clearly demonstrate the system functioning as designed, with melodic activity resulting in the shower turning on, and remaining on until the user ceases singing.

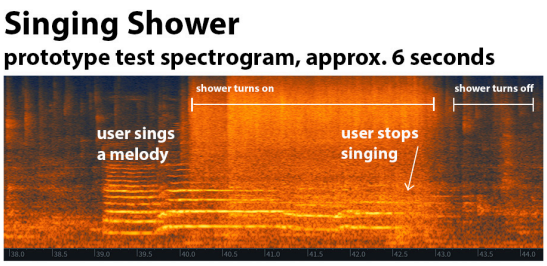


Figure 6. Spectrogram illustrating how the singing turn on the shower, while the absence of singing turns it off.

Further testing is required in order to fully evaluate the machine listening software. The tests in this paper were all conducted using fixed parameter values that were set when developing the system in the bathroom of the first researcher. It is likely that, for optimal performance, the system would need a calibration procedure when moved into another bathroom that takes into account the different shower system and room.

In the short term, a number of improvements are planned with regards to the hardware prototype. Notably, the powering system should be redesigned in order to control the solenoid valve from a battery, avoiding the need to plug the system into the wall. The solenoid valve should also be integrated into the enclosure for stability and ease of connection, and general improvements can be made to the containment system to make it more robust and usable inside the shower. A number of the failed attempts to activate the shower were also likely due to a low volume of singing, which could be addressed both with an automatic input gain system and with better physical placement of the microphone.

No matter the approach, we aim to carry out more systematic evaluation studies as the system is improved. Our tests showed that if a user is deliberately attempting to experiment with the system by using different sounds it can be difficult to assess whether to classify the resulting flow of water (or lack thereof) as a success or a failure. Similarly, some system failures are more frustrating or noticeable than others - if the water does not turn on when expected, for example, this is can be considerably more frustrating than if the water turns off unexpectedly.

With regards to the melodic sensing system offering a more privacy-sensitive approach to machine listening, there remains a challenge in terms of user experience design. This was illustrated by the user who voiced some concerns around possibly being recorded in the shower, despite the system being designed specifically to avoid that

approach. Any future system will need to account for existing negative user associations with audio surveillance.

Perhaps most importantly, real-world testing is required, with an improved hardware prototype installed in a home, and with participants tasked with using the system to take a shower. Through a study of this kind, we will be able to establish whether the enjoyment of the experience is long-lasting, whether some people would enjoy the system more than others (for example, children more than adults), whether there is an effect on energy consumption, and whether the experience increases awareness about energy consumption in the household. However transferring the testing to a home environment is a major challenge as they are of course deeply private spaces [42], and showering is a particularly intimate activity. The improved and more robust prototype will be designed with this type of in-home testing in mind.

7. FUTURE WORK

The tests described in this paper were relatively easy to set up due to the simple and standardized nature of the showers both in the author's home and within the KTH department, with a mixer containing a single 1/2-inch pipe fitting. A larger scale test would likely need to consider different types of showers, and how a prototype could accommodate an array of different fittings.

In the future we might consider including an AI-based approach [43] to the machine listening system, with a model built on a large corpus of recordings of people singing. This could improve the accuracy of the singing detection, however, it may bring back some of the ethical and privacy issues we aim to avoid.

Overall, this prototype represents the first output of a larger project looking at sonic interactions in the home, and how they can intersect with issues of energy consumption and resource efficiency. Whilst the relationship between singing and showering is particularly clear, we plan on continuing to explore the wider usability of melodic sound as a method for control in the domestic environment, as well as the use of a physical metaphor representing energy use.

As this project forms a part of a larger set of research regarding resource use, it will also be crucial to look more holistically at the overall benefits of introducing a new technology to mediate the use of hot water. Does, for example, the inclusion of a new electronic device (with associated lithium ion battery, silicon board, integrated circuits, etc) outweigh the benefits of potentially lower hot water use? The prototype described in this paper is still early stage and it was therefore not appropriate to attempt this analysis yet, but it should form a part of the review of any device that is made ready for more intensive testing. An analysis of this kind could draw inspiration from comparative life cycle assessments in projects like the Solar Powered Website [44].

8. CONCLUSION

We have described a project that brings together playful design methods, physical and sonic interaction design, and energy efficiency considerations. It does this by leveraging an existing common domestic behaviour in a surprising way. The results reported here of the evaluation of the initial prototype are promising, and we have outlined how we intend to develop the project in the future.

Whilst it is tempting to consider the potential viability of the Singing Shower as a product that could be installed on a wider scale, it is also worth acknowledging that it will likely encounter deeply entrenched resistance for a number of reasons. It is an interface that requires more physicality and active engagement than normal, all in the service of providing an admittedly less luxurious experience in the shower (a moment of a day which is often seen as special and relaxing). In this sense the project functions similarly to the "threshold devices" described by Michael and Gaver [45] by providing a physical metaphor that can open up discussions about resource use and our relationship to comfort and luxury. We consider this prototype, and the future iterations planned, both as a viable product which combines accessibility, usability, and interaction design considerations with sustainability issues, as well as a form of speculative and provocative design which challenges conventional framings of home energy use [46].

Acknowledgments

This work is carried out in the context of the "Sound for Energy" research project [7] funded by Swedish Energy Agency (Project No. 51645-1)

9. REFERENCES

[1] T. Hargreaves, M. Nye, and J. Burgess, "Making energy visible: A qualitative field study of how householders interact with feedback from smart energy monitors," *Energy policy*, vol. 38, no. 10, pp. 6111–6119, 2010.

[2] D. Lockton, F. Bowden, and C. Matthews, "Powerchord: Exploring ambient audio feedback on energy use," in *Living Labs*. Springer, 2017, pp. 297–308.

[3] D. Lockton, D. Harrison, and N. Stanton, "Making the user more efficient: Design for sustainable behaviour," *International journal of sustainable engineering*, vol. 1, no. 1, pp. 3–8, 2008.

[4] B. Asare-Bediako, W. Kling, and P. Ribeiro, "Home energy management systems: Evolution, trends and frameworks," in *2012 47th International Universities Power Engineering Conference (UPEC)*. IEEE, 2012, pp. 1–5.

[5] H. Allcott, "Social norms and energy conservation," *Journal of public Economics*, vol. 95, no. 9-10, pp. 1082–1095, 2011.

[7] `https://soundforenergy.net`

[6] T. Hargreaves, M. Nye, and J. Burgess, "Keeping energy visible? exploring how householders interact with feedback from smart energy monitors in the longer term," *Energy policy*, vol. 52, pp. 126–134, 2013.

[7] Y. Strengers, "Smart energy in everyday life: are you designing for resource man?" *interactions*, vol. 21, no. 4, pp. 24–31, 2014.

[8] P. Slovic and E. Peters, "Risk perception and affect," *Current directions in psychological science*, vol. 15, no. 6, pp. 322–325, 2006.

[9] C. Oltra, A. Boso, J. Espluga, and A. Prades, "A qualitative study of users' engagement with real-time feedback from in-house energy consumption displays," *Energy Policy*, vol. 61, pp. 788–792, 2013.

[10] J. Pierce, C. Fan, D. Lomas, G. Marcu, and E. Paulos, "Some consideration on the (in) effectiveness of residential energy feedback systems," in *Proceedings of the 8th ACM conference on designing interactive systems*, 2010, pp. 244–247.

[11] S. Darby, "Smart metering: what potential for householder engagement?" *Building research & information*, vol. 38, no. 5, pp. 442–457, 2010.

[12] D. Lockton, G. Crawford, D. Singh, and S. Wu, "Electric acoustic: Exploring energy through sonic & vibration displays," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–6.

[13] K. Groß-Vogt, M. Weger, R. Höldrich, T. Hermann, T. Bovermann, and S. Reichmann, "Augmentation of an Institute's Kitchen: An Ambient Auditory Display of Electric Power Consumption," in *Proceedings of the 24th International Conference on Auditory Display - ICAD 2018*. The International Community for Auditory Display, 2018, pp. 105–112. [Online]. Available: http://hdl.handle.net/1853/60088

[14] P. Cowden and L. Dosiek, "Auditory displays of electric power grids," in *Proceedings of the 24th International Conference on Auditory Display - ICAD 2018*. The International Community for Auditory Display, 2018, pp. 113–120.

[15] S. Pauletto, H. Cambridge, and R. Rudnicki, *Introduction to SoniHED Conference Proceedings*. United Kingdom: University of York, 2014.

[16] S. Pauletto and R. Bresin, "Sonification research and emerging topics," in *Doing Research in Sound Design*. Focal Press, 2021, pp. 238–254.

[17] M. Houel, A. Arun, A. Berg, A. Iop, A. Barahona-Rios, and S. Pauletto, "Perception of emotions in knocking sounds: An evaluation study," in *Sound and Music Computing Conference 2020, Torino, 24-26 June 2020*, 2020.

[18] B. L. Giordano, H. Egermann, and R. Bresin, "The production and perception of emotionally expressive walking sounds: Similarities between musical performance and everyday motor activity," *PLoS One*, vol. 9, no. 12, p. e115587, 2014.

[19] S. Connor, *Beyond words: Sobs, hums, stutters and other vocalizations*. Reaktion Books, 2014.

[20] G. Oleksik, D. Frohlich, L. M. Brown, and A. Sellen, "Sonic interventions: understanding and extending the domestic soundscape," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, 2008, pp. 1419–1428.

[21] D. M. Shull, "Water is the new oil: The national security implications of our looming water crisis," *Naval L. Rev*, vol. 65, p. 117, 2016.

[22] V. Peart, "The Conservation Balancing Act: Part II. In the Bathroom," *FCS3233, University of Florida, Institute of Food and Agricultural Sciences*, p. 4, 2001. [Online]. Available: http://ufdcimages.uflib.ufl.edu/IR/00/00/21/25/00001/FY13800.pdf

[23] M. Pomianowski, H. Johra, A. Marszal-Pomianowska, and C. Zhang, "Sustainable and energy-efficient domestic hot water systems: A review," *Renewable and Sustainable Energy Reviews*, vol. 128, p. 109900, Aug. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1364032120301921

[24] I. Ibn Khaldun, N. J. Dawood, B. B. Lawrence, and F. Rosenthal, *The Muqaddimah: An Introduction to History - Abridged Edition*, 2020, oCLC: 1153486831. [Online]. Available: https://www.degruyter.com/isbn/9781400866090

[25] J. Huizinga, *Homo Ludens - A Study of the Play Element in Culture*. Beacon Press, 1950.

[26] W. Gaver, "Designing for homo ludens," *I3 Magazine*, vol. 12, no. June, pp. 2–6, 2002.

[27] W. W. Gaver, J. Bowers, K. Boehner, A. Boucher, D. W. Cameron, M. Hauenstein, N. Jarvis, and S. Pennington, "Indoor weather stations: investigating a ludic approach to environmental HCI through batch prototyping," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 3451–3460.

[28] C. Mivielle, "What is ludic about ludic design? a back and forth between theory and practice," in *Proceedings of 11th European Academy of Design Conference*, 2015.

[29] W. J. Fitzgerald, "VOTAN V5000 speech recognition system performance test report," Final Report, Mar. - May 1984 Naval Ocean Systems Center, San Diego, CA., Aug. 1984.

[30] K. H. Sørensen and M. Lie, *Making technology our own? Domesticating technology into everyday life*. Oslo: Scandinavian University Press, 1996.

[31] T. Hargreaves, C. Wilson, and R. Hauxwell-Baldwin, "Learning to live in a smart home," *Building Research & Information*, vol. 46, no. 1, pp. 127–139, Jan. 2018. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/09613218.2017.1286882

[32] R. Anuradha, A. B. Kocaballi, I. Nicenboim, M. L. J. Sondergaard, M. L. Lupetti, C. Key, C. Speed, D. Lockton, E. Giaccardi, F. Gromme *et al.*, "Making everyday things talk: Speculative conversations into the future of voice interfaces at home," in *CHI conference on Human Factors in Computing Systems*. Association for Computing Machinery (ACM), 2021.

[33] N. J. Thrift, *Non-representational theory: space, politics, affect*, ser. International Library of Sociology. Milton Park, Abingdon, Oxon ; New York, NY: Routledge, 2008.

[34] M. Ford and W. Palmer, "Alexa, are you listening to me? An analysis of Alexa voice service network traffic," *Personal and Ubiquitous Computing*, vol. 23, no. 1, pp. 67–79, Feb. 2019. [Online]. Available: http://link.springer.com/10.1007/s00779-018-1174-x

[35] E. Furey and J. Blue, "Alexa, Emotions, Privacy and GDPR," Jul. 2018. [Online]. Available: https://scienceopen.com/document?vid=f0762f59-c68a-44e7-85ba-64b21770178e

[36] S. Kennedy, H. Li, C. Wang, H. Liu, B. Wang, and W. Sun, "I Can Hear Your Alexa: Voice Command Fingerprinting on Smart Home Speakers," in *2019 IEEE Conference on Communications and Network Security (CNS)*. Washington DC, DC, USA: IEEE, Jun. 2019, pp. 232–240. [Online]. Available: https://ieeexplore.ieee.org/document/8802686/

[37] M. Tanabian and R. Goubran, "Accent adaptation in speech user interface," in *IEEE Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2005.* Messina, Italy: IEEE, 2005, p. 4. [Online]. Available: http://ieeexplore.ieee.org/document/1567572/

[38] W. Odom, R. Wakkary, Y.-k. Lim, A. Desjardins, B. Hengeveld, and R. Banks, "From Research Prototype to Research Product," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. San Jose California USA: ACM, May 2016, pp. 2549–2561. [Online]. Available: https://dl.acm.org/doi/10.1145/2858036.2858447

[39] G. Wallenborn and H. Wilhite, "Rethinking embodied knowledge and household consumption," *Energy Research & Social Science*, vol. 1, pp. 56–64, 2014.

[40] H. D. Shin and T. Bhamra, "Design for sustainable behaviour: a case study of using human-power as an everyday energy source," *Journal of Design Research*, vol. 14, no. 3, pp. 280–299, 2016.

[41] B. Laugwitz, T. Held, and M. Schrepp, "Construction and evaluation of a user experience questionnaire," in *Symposium of the Austrian HCI and usability engineering group*. Springer, 2008, pp. 63–76.

[42] W. Gaver, J. Bowers, A. Boucher, A. Law, S. Pennington, and B. Walker, "Electronic Furniture for the Curious Home: Assessing Ludic Designs in the Field," *International Journal of Human-Computer Interaction*, vol. 22, no. 1-2, pp. 119–152, Apr. 2007. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10447310709336958

[43] R. Monir, D. Kostrzewa, and D. Mrozek, "Singing Voice Detection: A Survey," *Entropy*, vol. 24, no. 1, p. 114, Jan. 2022. [Online]. Available: https://www.mdpi.com/1099-4300/24/1/114

[44] R. R. Abbing, "'This is a solar-powered website, which means it sometimes goes offline': a design inquiry into degrowth and ICT," *LIMITS Workshop on Computing within Limits*, Jun. 2021. [Online]. Available: https://limits.pubpub.org/pub/lecuxefc

[45] M. Michael and W. Gaver, "Home Beyond Home: Dwelling With Threshold Devices," *Space and Culture*, vol. 12, no. 3, pp. 359–370, Aug. 2009. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1206331209337076

[46] T. Hargreaves, "Beyond energy feedback," *Building Research & Information*, vol. 46, no. 3, pp. 332–342, Apr. 2018. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/09613218.2017.1356140

Design Concepts for Gaze-Based Digital Musical Instruments

Nicola Davanzo
Laboratory of Music Informatics (LIM)
Department of Computer Science
University of Milan
nicola.davanzo@unimi.it

Federico Avanzini
Laboratory of Music Informatics (LIM)
Department of Computer Science
University of Milan
federico.avanzini@unimi.it

ABSTRACT

In recent decades, the introduction of new affordable non-invasive eye tracking technologies accelerated the development of new gaze interaction techniques. Some of these find an application in accessible human-computer interfaces operable by people with quadriplegic disabilities. Musical interfaces represent a possible benchmark for these techniques since they require high precision levels, speed, and minimal latency. Several software Accessible Digital Musical Instruments have been developed, experimenting with keys and visual cues suitable for this particular context. This paper proposes a review of different design techniques proposed in the literature for the design of gaze-based musical interfaces, as well as possible solutions for the Midas Touch Problem, a known issue in gaze-based interfaces. A summary of the physiology of the human ocular movement is also provided. The provided notions could inform the design of new gaze-based software musical instruments.

1. INTRODUCTION

Research on Digital Musical Instruments (or DMIs) has expanded over the past decades into the use of unconventional interfaces, interaction paradigms and channels. Such instruments are less constrained by physical limitations than their acoustic and traditional counterparts: this allowed for the exploration of new expressive possibilities and led to the need of partially revising what we culturally consider a musical instrument.

One of the possibilities offered by DMIs is to increase the accessibility of the world of musical performance, extending it to people with important motor disabilities such as quadriplegic users, which are paralyzed from the shoulders below, leaving only the neck and above, with all the related possible bodily movements as available interaction channels. The use of gaze as an interaction channel for musical interfaces can be a particularly simple and suitable solution for designing Accessible Digital Musical Instruments (ADMIs) dedicated to this particular target. This is also demonstrated by the amount of gaze-based accessible instruments found in the literature.

Copyright: © 2022 Nicola Davanzo et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

A 2019 ADMIs review of by Frid [1] shows that among the 83 instruments found in literature, dedicated to different types of user groups, disabilities and contexts, 2-3% exploited a gaze-based interface.

Gaze detection is now an established interaction method for different contexts in Human-Computer Interaction [2, 3]. This was also possible thanks to the introduction on the market of several low-cost eye trackers, such as Tobii (EyeX, 4C, 5) [1] and The EyeTribe devices (now discontinued). These eye trackers take advantage of non-invasive technologies based on Near Infrared (NIR) cameras that require a straightforward calibration procedures, whose detection performances have been evaluated in the past [4,5]. In this paper we refer to the characteristics of these eye trackers to describe interface design techniques and related interaction issues.

In a previous article we listed gaze as one of the possible interaction channels for ADMIs dedicated to quadriplegic users [6], then we carried out a comparison experiment between gaze tracking, head movement, breath and mouse in target selection tasks [7]. Compared to the others, gaze resulted to be a particularly fast interaction channel, although generally NIR eye trackers present unstable detection and noise.

The interfaces of acoustic musical instruments are usually designed to exploit the peculiarities of hand and finger movements. Similarly, a gaze-based musical interface should consider the characteristics of eye movements to guarantee comfortable and effective interaction. In fact, a simple imitation of the layout of a traditional musical instrument may be unsuitable for gaze interaction, thus requiring the implementation of specific solutions.

The goal of this paper is to provide theoretical background on design choices and techniques for gaze-based musical interfaces design, as well as a collection of related design cues. Many of these have been found and implemented by instruments found in literature and resumed here.

Some words should be spent to indicate that, despite some of the theoretical notions we're listing could be useful in designing any type of gaze based DMIs, focus is given to a particular category of instruments dedicated to real-time performance. In order to frame this category, we resort to Malloch *et al.* [8] conceptual framework, which classifies DMIs according to which kind of behavior is re-

[1] Tobii products on Tobii official website: *https://tech.tobii.com/products/*

quired to interact with them. In such framework, behaviors can be skill-, rule-, or model-based. Playing conventional acoustic instruments require skill-based behaviors, i.e. actions which take place partly without conscious effort, in an automated manner due to muscles memory training, to which the instrument immediately responds with feedback (e.g. acoustic) which impacts the performed music. In this paper we're focusing on this type of instruments, while rule-based or model-based gaze controlled musical interfaces (such as turntables, sequencers, algorithmic music composition, etc.) are instead not directly covered.

Sec. 2 provides an insight on the related state of the art. Sec. 3 describes how the eyes move from a physiological point of view. Sec. 4 lists a series of design cues and techniques which could be used to enhance interaction. Finally, Sec. 5 addresses the Midas Touch problem, a known issue in gaze-based interfaces design, and some possible solutions to tackle it.

2. RELATED WORKS

Relatively few gaze-based methods for playing music have been developed to date. Interesting analyses of strengths and weaknesses of these approaches, as well as limits and challenges that future solutions should address, can be found in the works by Hornof et al. [9, 10]. Vamvakousis [11] also provides a source for gaze-based instruments research.

Up to date, a few gaze-based instrument proposed different solutions for gaze-based musical performance. The EyeMusic system and related performances [12] are a first attempt at creating tools for generating sounds with the eyes, although they cannot be strictly considered real musical instruments. The Eye Play The Piano intrument [13] allows to select notes and chords by looking at hexagonal graphical shapes that control the keys of a real piano. EyeJam [14] proposes a method for note selection called "context-switch", where sound is produced only when the gaze crosses a horizontal line. Lumiselo [15], is probably the first to propose a hybrid method involving both gaze and breath (through a sip-and-puff controller): a note is selected by gaze, and then its actual playing occurs by blowing into a breath detecting sensor. Netytar [16, 17] exploited the same idea of hybrid gaze-breath interaction, focusing on proposing a peculiar isomorphic layout designed to solve gaze interaction issues and providing fast detection through the avoidance of eye tracker data filtering. Netychords [18] used instead a gaze-head hybrid interaction paradigm to control chords performance. EyeConductor [19], and The EyeHarp [20] introduced pie-shaped interfaces in which the central area is mapped to silence (pauses). EyeConductor also exploits facial expressions, such as raising eyebrows or opening the mouth to change octaves or to control filters. The EyeHarp is a complete musical instrument that allows to play notes on several octaves and to control sound dynamics. Its interface contain gaze sensitive buttons with white dots used to guide the gaze, and the central area is exploited for both pauses and note repetition, through a dynamically mapped button. Clarion [21] instead proposes a solution based on customizable layouts. In this way, the interface could be customized according to the musical piece to be performed, renouncing however a general consistency that could affect the instrument learning process.

The design solutions provided by these instruments are reviewed, collected and compared in the following sections.

3. CHARACTERISTICS OF GAZE MOVEMENTS

Gaze point is the point in space (or the point on the screen) where the user is looking at.

Eyes generally move through *saccades*, which are jerky movements, lasting about 30 ms, during which the gaze point moves from one discrete point to another. These are interspersed with *fixations*, where the gaze point remains, indeed, almost fixed on a position. Usually a fixation lasts from 100 ms to 400 ms. Fig. 1 shows a visual representation of gaze point moving through saccades and fixations while reading a text.

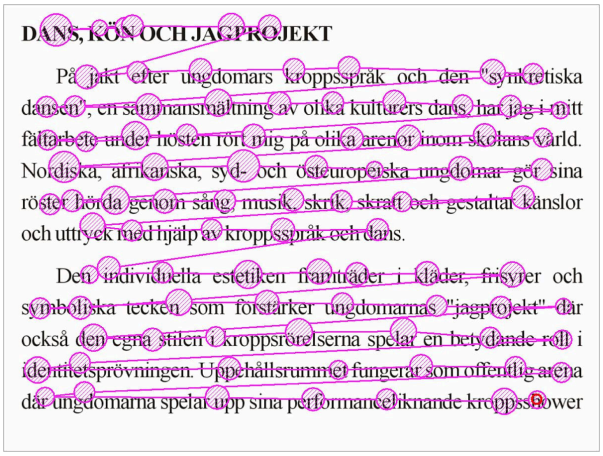


Figure 1: A *scanpath*, namely a visual representation of saccades (straight lines) interspersed with fixations (circles) while reading a text
Source: *Lucs-kho at English Wikipedia, Public domain, via Wikimedia Commons*

That said, the eye is unable to perform fluid movements unless it has a target to lock on: this is called *smooth pursuit*, a fluid movement which follows the movement of a target.

Blinks are sometimes not recommended as an interaction channel due to their potentially involuntary nature [2], but are listed in [6] as one of the residual movement abilities for a quadriplegic person. As they are very fast, however, blinks are employed in some accessible applications and instruments like Netytar [16], using some filter or rule to discriminate voluntary and involuntary blinks.

Finally, even during a fixation eyes are not perfectly still but make small random movements within 0.1° of the visual angle, called *jitter*.

These movements can be activated voluntarily, but many can occur involuntarily and unconsciously. Involuntary saccades, for example, occur on a regular basis even during

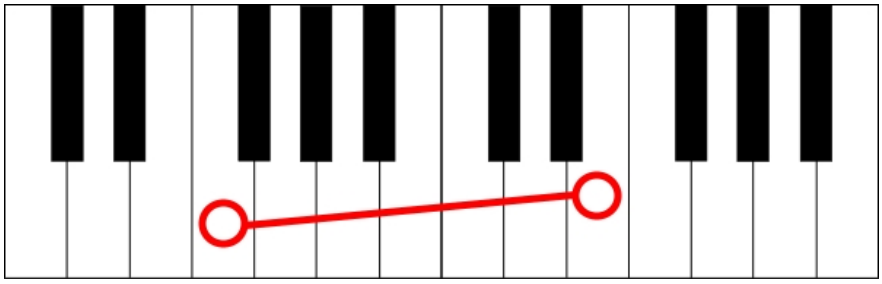


Figure 2: Gaze scanpath on a piano keyboard. When gaze moves from the F key to the E key, intermediate keys are crossed.

fixations [22]. Those may preclude musical performance, which requires very precise control. There is evidence for gaze anticipating physical movement [23] and interactions in virtual environments [24], a behavior which the performer must learn to avoid during gaze-controlled musical performance. In gaze controlled instruments, those may lead to the anticipated performance of notes with respect to the prescribed tempo, unless the introduction of filters to compensate by creating latency. Such behavior was noticed during the evaluation of The EyeHarp [20, Sec. 2.2.2]. Instruments like EyeJam, Netytar and Netychords, as a counterexample, does not use filters in order to improve the precision at higher tempos [16], thus not providing any aid to avoid anticipations.

The rhythmic capabilities of eye movements are limited. Hornof [10] provided an eye-tapping experiment which shows that eyes are unable to deliberately perform more than 4 saccadic movements per second (approximately one saccade every 250 ms). According to the author, this seems to be an upper limit which cannot be overcome, not even through training. In systems where notes are selected through gaze pointing, this translates into a maximum limit in note changing speed. More trained people could however manage to maintain fast tempos with greater precision. In [17] we discussed and proposed a training method to possibly reach this goal through exercising.

4. VISUAL CUES AND TECHNIQUES

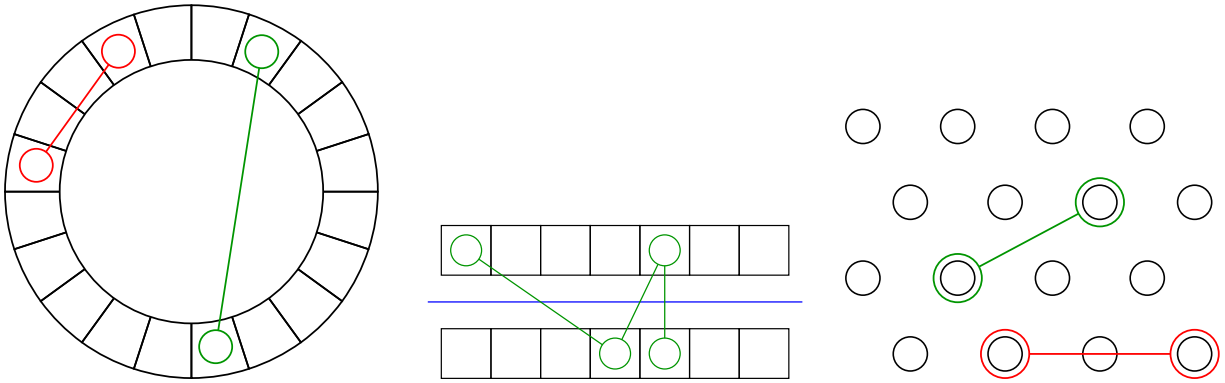
The following are some techniques which can be considered and combined while designing a gaze-based musical interface.

Color. When using a gaze-based musical interface, an eye movement can result in a involuntary interaction. This leaves little space for the user to explore the interface, and usually the performer needs to know in advance where the next gaze movement should happen. While many musical interfaces employ differently shaped keys (e.g. a normal piano keyboard) or spatialization (e.g. in The EyeHarp's interface [20]) to help note localization, color can be used strategically to enhance interaction and partially solve this problem. It has been proven that the areas of sight outside the fovea (the central area of human vision), corresponding to peripheral vision, are particularly sensitive to contrasting color variations [25].

Cursors. Although showing a cursor is a classical way to give a visual feedback to the user for the current pointing position, its usage in gaze-based interfaces could be problematic since it could distract the user. It has been shown, through experiments on primates, that involuntary gaze movements can be caused by moving objects [26]. Furthermore, given the general imperfect accuracy and precision of eye tracker data, even a slightly different position of the cursor with respect to the fixation point could frustrate the user and feel unnatural. It can be argued that the use of visual feedback may not be necessary to indicate the user's gaze position. When using a pointing device such as a mouse, a cursor is required as the pointed position would otherwise not be known to the user. In the case of gaze, the position is already known since the user who knows where their gaze point lies. There are however alternatives to cursors to return visual feedback on selecting items: one of these is to highlight the selected element through a different color, a flash or a different shape when the gaze point enters its area. Alternatively, it is possible to implement "discrete cursors" (e.g. as the one proposed by Netytar's interface [16]), which instead of moving in a continuum can only assume a limited number of positions (e.g. centered on gaze sensitive elements).

Visual elements to enhance precision. Given the aforementioned jittering nature of eye movements and gaze detection by eye trackers, some visual elements can be introduced to enhance interaction precision. Other than increasing the dimension of the gaze sensitive elements, they can be equipped with "visual hooks", such as dots, to help the user concentrate fixations on the center of their area. The EyeHarp [20], for example, presents a series of points on keys and external areas (which are used for pauses).

Auto-scrolling. Netytar [16] and Netychords [18] introduced an auto-scrolling feature and approach. The view on the virtual keyboard moves automatically, so the currently gazed key is smoothly placed on the center of the visualization area. The speed at which the interface moves is proportional to the square of the distance between the observed point and the center of visualization area on screen. This allows to have a theoretically infinite playing region available, regardless of screen size. This solution could also increase detection accuracy, as gaze detection provided by eye trackers based on Near Infrared technology is usually more accurate in the central screen area [27].



(a) Pie-shaped layout (b) Context switch based layout (c) Layout presenting spacing between keys

Figure 3: Three possible keys layouts to tackle the Midas Touch problem. In a pie-shaped layout (a), the red colored scanpath crosses intermediate keys, while the green colored one does not. In a context switch based layout (b), the green colored scanpath does not cross intermediate keys, since it passes through the blue rows separator in the middle. Spacing between keys (c) can solve the problem for some intervals (green scanpath) but not for others (red scanpath).

5. THE MIDAS TOUCH PROBLEM

One of the most known issues to be addressed in the creation of gaze based interfaces, occurring also while designing other types of interfaces based for example on gestural controls, is the so-called "Midas Touch" problem [3, 28, 29]. It consists of the fact that, if the act of passing gaze point through the area of on an interface element triggers an event such as its activation (as often happens in gaze-based musical interfaces), any exploratory or involuntary gaze movement can potentially cause an unwanted interaction. Jacob [2] summarizes the problem with the following sentence:

"Everywhere you look, another command is activated; you cannot look anywhere without issuing a command."

One very important consequence in musical interfaces is that keys layout design is a non-trivial problem which requires an additional effort. Traditional acoustic musical instrument layouts may not be suitable for gaze-based interaction. Let us take as an example a piano keyboard. In order to perform any given musical interval which requires a jump between two non-adjacent keys, other keys should be crossed (Fig. 2). Even if a saccadic movement is very fast, the sampling frequency of modern eye trackers is high enough to detect intermediate positions, causing an involuntary activation of intermediate keys. While fingers can be lifted from a keyboard, it is not possible to control the musical performance in the same way with gaze.

Various solutions have been proposed to this problem in the literature, for both musical and general purpose gaze-based interfaces.

Dwell time. A possible solution to this problem is to apply a delayed selection method. Using dwell time, an interface element is selected by gaze entering its area, but activated after the expiration of a given time interval [30, 31]. In musical terms, however, this might not be a very efficient solution since it introduces a Delayed Auditory Feedback (DAF) between the action of the physical input and the generation of the related sound. A DAF may alter the quality of a musical performance, impeding correct play of rhythmic pieces [32]. According to Wessel and Wright [33], 10 ms are an acceptable upper bound for a delay on audible system reactions during live computer music performances.

Filtering. Another solution is applying a filter to discriminate saccadic movements from fixations, and enable activations only when a fixation occurs. An implementation is provided by The EyeHarp [20]. However, even in The EyeHarp a DAF was observed which could preclude the performance of rapid sequences of notes [16].

Hybrid interaction. Using an extra physical channel in addition to gaze allows to decouple the note selection from its performance. As an example, the already cited Lumiselo [15] and Netytar [16] exploit breath to control note onsets and sound intensity: when no breath is emitted, gazed keys are not activated. Netychords [18] exploits instead head rotation along the yaw axis to trigger chord strummings and control sound intensity. It has to be noted, however, that introducing further interaction methods In addition to gaze could hinder the usability of the instrument to users with more restricted motor capabilities (such as those affected by total locked-in syndrome [34].

Keys displacement. Passing through intermediate keys during the performance of different musical intervals can be avoided, in part or completely, through an adequate keys positioning. Various solutions have been proposed in literature, all having pros and cons, being partially capable of solving the problem.

- **Pie shaped layouts**. A layout where the keys are arranged in a circular fashion, as illustrated in Fig. 3a, can partially solve the problem. As shown in the figure, ideally many musical intervals do not require crossing keys in between. However, with finite-sized keys, some intervals may still require passing through intermediate keys. One drawback of using this solution is that the space is not fully exploited and, since the eye tracker detection limitations re-

Problem	Solutions
(3) *Involuntary movements (saccades, jitter)*	?
(3) *Anticipated performance of notes*	Introduction of filters to create latency
(3) *Limited rhythmic capabilites of the eyes*	?
(4) *Impossibility to explore the interface without issuing commands*	Strategic use of color to exploit color sensitivity outside fovea
(4) *Cursors can be distracting and suffer eye tracker's inaccuracy*	Highlight the gazed elements with color, flashes, shape changes; Use a "discrete cursor"
(4) *Eye movements and eye tracker's detection is jittery*	Equipping gaze sensitive elements with "visual hooks" (e.g. in the center of the area); Enlarging the gaze sensitive areas
(4) *Limited dimension of the screen*	Introduction of auto-scrolling, exploiting smooth pursuit
(5) *Midas Touch problem*	Dwell time (which introduces DAF); Fixation discrimination filters (which introduce DAF); Hybrid interaction (gaze + another interaction channel); Keys displacement (pie shaped layouts, context switching, spacing between keys).

Table 1: A list of problem treated in this paper with relative solutions. Numbers between parentheses indicate the section in this paper where the problem and the solutions are investigated. Question marks denote actually open challenges.

quire the use of large keys, it is not possible to represent more than a given number of notes on the screen. It is also not easy to implement solutions such as auto-scrolling (Sec. 4) for this type of layouts. The aforementioned Eye Conductor [19] and EyeHarp [20] make use of pie-shaped layouts.

- **Context switching**. A particular and original solution, called "context switching", has been proposed by the instrument EyeJam [14]. The solution, summarized by Fig. 3b, consists in placing two rows (or columns) of keys mapped to the same notes. The two rows should be separated by an area non-sensitive to gaze. Any activation should be preceded by the act of crossing this area. Any gaze movement which does not follow this rule is substantially ignored. In this way, an up-and-down motion of the gaze is required, but intermediate keys crossing is avoided.
- **Spacing between keys**. Another solution to tackle this problem amounts to using a 2D layout where keys are interspersed with non-sensitive areas and placed in a strategic way to avoid intermediate keys crossing for common musical intervals. This is obtained by reducing the size of keys and/or setting the gaze sensitive area associated with each key (often called "occluder") to have a different dimension than the key itself, as for example happens in Netytar [16], whose layout design is exemplified in Fig. 3c.

6. DISCUSSION AND CONCLUSIONS

In this paper we have summarized techniques and cues found in literature which can inform the design of gaze-based musical instruments.

Table 1 presents a list of problems and solutions addressed in this paper. For some issues, a solution is yet not present in literature, or solutions are partial and not sufficient. Those have been denoted with a question mark in the table.

Involuntary eye movements for example are part of the physiology of gaze movement, but can present an additional difficulty to take into account while designing gaze-based interfaces. Experiments like Eye Music from Hornof *et al.* [12, 35] circumnavigate this problem by sonifying these movements and making them part of the musical experience.

Although eyes move fastly, saccadic movement frequency limits can hinder the performance of fast pieces and virtuosities, and no solutions have been proposed so far to stem this issue, leaving another question mark on the "limited rhythmic capabilities of the eyes" problem. Although training could help to reach the physiological limit, solutions to map multiple effects to a single eye movement (e.g. a sequence of notes) could be introduced.

Some interaction problems that have been tackled in previous remains unresolved, keeping related discussions and the need to develop new techniques open. As an example, although several solutions have been summarized to tackle the Midas Touch problem, each of them involves different limitations on the note keys layout. Noise introduced by the image-based nature of the eye tracking hardware involves the need to use filters, which introduce input lag, or larger keys, which reduce the number of displayed gaze-sensitive elements.

Since musical interactions require both high spatial accuracy and temporal resolution, it results to be a very demanding testing ground for gaze-based interaction techniques. A solution for these common problems could allow the development of gazed-based interfaces suitable for different applications in the general areas of accessibility and Human-Computer Interaction.

7. REFERENCES

[1] E. Frid, "Accessible Digital Musical Instruments—A Review of Musical Interfaces in Inclusive Music Practice," *Multimodal Technologies and Interaction*, vol. 3, no. 3, p. 57, Jul. 2019.

[2] R. J. K. Jacob, "Eye tracking in advanced interface design," in *Virtual Environments and Advanced Interface Design*. USA: Oxford University Press, Inc., 1995, pp. 258–288.

[3] P. Majaranta and A. Bulling, "Eye Tracking and Eye-Based Human–Computer Interaction," in *Advances in Physiological Computing*, ser. Human–Computer Interaction Series, S. H. Fairclough and K. Gilleade, Eds. London: Springer, 2014, pp. 39–65.

[4] G. Funke, E. Greenlee, M. Carter, A. Dukes, R. Brown, and L. Menke, "Which Eye Tracker Is Right for Your Research? Performance Evaluation of Several Cost Variant Eye Trackers," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 60, no. 1, pp. 1240–1244, Sep. 2016.

[5] S. Popelka, Z. Stachoň, Č. Šašinka, and J. Doležalová, "EyeTribe Tracker Data Accuracy Evaluation and Its Interconnection with Hypothesis Software for Cartographic Purposes," *Computational Intelligence and Neuroscience*, vol. 2016, p. e9172506, Mar. 2016.

[6] N. Davanzo and F. Avanzini, "Hands-Free Accessible Digital Musical Instruments: Conceptual Framework, Challenges, and Perspectives," *IEEE Access*, vol. 8, pp. 163 975–163 995, 2020.

[7] ——, "Experimental Evaluation of Three Interaction Channels for Accessible Digital Musical Instruments," in *Proc. '20 Int. Conf. on Computers Helping People With Special Needs*. Online Conf.: Springer, Cham, Sep. 2020, pp. 437–445.

[8] J. Malloch, D. Birnbaum, E. Sinyor, and M. M. Wanderley, "Towards a New Conceptual Framework for Digital Musical Instruments," in *Proc. 9th Int. Conf. on Digital Audio Effects*, Montreal, Canada, September 18-20, 2006, 2006, pp. 49–52.

[9] A. Hornof, T. Rogers, J. Stolet, and T. Halverson, "Bringing to Life the Musical Properties of the Eyes," University of Oregon Department of Computer and Information Science, Technical Report 08-05, 2008.

[10] A. J. Hornof, "The Prospects For Eye-Controlled Musical Performance," in *Proc. 14th Int. Conf. on New Interfaces for Musical Expression (NIME'14)*, ser. NIME 2014, Goldsmiths, University of London, UK, Jul. 2014.

[11] Z. Vamvakousis, "Digital Musical Instruments for People with Physical Disabilities," Ph.D. dissertation, Universitat Pompeu Fabra Barcelona, 2016.

[12] A. J. Hornof, T. Rogers, and T. Halverson, "EyeMusic: Performing live music and multimedia compositions with eye movements," in *Proceedings of the 7th International Conference on New Interfaces for Musical Expression*, ser. NIME '07. New York, NY, USA: Association for Computing Machinery, Jun. 2007, pp. 299–300.

[13] Fove Inc., "Eye Play the Piano," http://eyeplaythepiano.com/en/, 2020.

[14] C. H. Morimoto, A. Diaz-Tula, J. A. T. Leyva, and C. E. L. Elmadjian, "Eyejam: A Gaze-Controlled Musical Interface," in *Proceedings of the 14th Brazilian Symposium on Human Factors in Computing Systems*, ser. IHC '15. Salvador, Brazil: ACM, 2015, pp. 37:1–37:9.

[15] S. Bailey, A. Scott, H. Wright, I. M. Symonds, and K. Ng, "Eye.Breathe.Music: Creating music through minimal movement," in *Proc. Conf. Electronic Visualisation and the Arts (EVA 2010)*, London, UK, Jul. 2010, pp. 254–258.

[16] N. Davanzo, P. Dondi, M. Mosconi, and M. Porta, "Playing music with the eyes through an isomorphic interface," in *Proc. of the Workshop on Communication by Gaze Interaction*. Warsaw, Poland: ACM Press, 2018, pp. 1–5.

[17] N. Davanzo and F. Avanzini, "A Method for Learning Netytar: An Accessible Digital Musical Instrument:," in *Proceedings of the 12th International Conference on Computer Supported Education*. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, 2020, pp. 620–628.

[18] N. Davanzo, M. De Filippis, and F. Avanzini, "Netychords: An Accessible Digital Musical Instrument for playing chords using gaze and head movements," in *In Proc. '21 Int. Conf. on Computer- Human Interaction Research and Applications (CHIRA '21)*, Online conf., 2021.

[19] A. Refsgaard, "Eye Conductor," https://andreasrefsgaard.dk/projects/eye-conductor/, 2021.

[20] Z. Vamvakousis and R. Ramirez, "The EyeHarp: A Gaze-Controlled Digital Musical Instrument," *Frontiers in Psychology*, vol. 7, p. article 906, 2016.

[21] OpenUpMusic, "The Clarion," http://openupmusic.org/the-clarion/, 2021.

[22] D. Purves, G. J. Augustine, D. Fitzpatrick, L. C. Katz, A.-S. LaMantia, J. O. McNamara, and S. M. Williams, "Types of Eye Movements and Their Functions," *Neuroscience. 2nd edition*, pp. 361–390, 2001.

[23] B. Gesierich, A. Bruzzo, G. Ottoboni, and L. Finos, "Human gaze behaviour during action execution and observation," *Acta Psychologica*, vol. 128, no. 2, pp. 324–330, Jun. 2008.

[24] J. B. Badler and A. Canossa, "Anticipatory Gaze Shifts during Navigation in a Naturalistic Virtual Environment," in *Proc. of the 2015 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '15)*. London, United Kingdom: Association for Computing Machinery, Oct. 2015, pp. 277–283.

[25] C. I. Lou, D. Migotina, J. P. Rodrigues, J. Semedo, F. Wan, P. U. Mak, P. I. Mak, M. I. Vai, F. Melicio, J. G. Pereira, and A. Rosa, "Object Recognition Test in Peripheral Vision: A Study on the Influence of Object Color, Pattern and Shape," in *Proc. Int. Conf. on Brain Informatics*, ser. Lecture Notes in Computer Science, F. M. Zanzotto, S. Tsumoto, N. Taatgen, and Y. Yao, Eds. Berlin, Heidelberg: Springer, 2012, pp. 18–26.

[26] K. Guo and P. J. Benson, "Involuntary eye movements in response to first- and second-order motion," *NeuroReport*, vol. 9, no. 15, pp. 3543–3548, Oct. 1998.

[27] K. Holmqvist, M. Nyström, and F. Mulvey, "Eye tracker data quality: What it is and how to measure it," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '12. New York, NY, USA: Association for Computing Machinery, Mar. 2012, pp. 45–52.

[28] B. Velichkovsky, M. A. Rumyantsev, and M. A. Morozov, "New Solution to the Midas Touch Problem: Identification of Visual Commands Via Extraction of Focal Fixations," *Procedia Computer Science*, vol. 39, Dec. 2014.

[29] B. Velichkovsky, A. Sprenger, and P. Unema, "Towards gaze-mediated interaction: Collecting solutions of the "Midas touch problem"," in *Human-Computer Interaction INTERACT '97: IFIP TC13 International Conference on Human-Computer Interaction, 14th–18th July 1997, Sydney, Australia*, ser. IFIP — The International Federation for Information Processing, S. Howard, J. Hammond, and G. Lindgaard, Eds. Boston, MA: Springer US, 1997, pp. 509–516.

[30] J. P. Hansen, A. S. Johansen, D. W. Hansen, K. Ito, and S. Mashino, "Command Without a Click: Dwell Time Typing by Mouse and Gaze Selections," in *Proc. Int. Conf. on Human-Computer Interaction (INTERACT '03)*, M. Rauterberg, M. Menozzi, and J. Wesson, Eds. Zurich, Switzerland: IOS Press, 1.

[31] P. Majaranta, U.-K. Ahola, and O. Špakov, "Fast gaze typing with an adjustable dwell time," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: Association for Computing Machinery, Apr. 2009, pp. 357–360.

[32] P. Pfordresher and C. Palmer, "Effects of delayed auditory feedback on timing of music performance," *Psychological Research*, vol. 66, no. 1, pp. 71–79, Feb. 2002.

[33] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," *Computer Music J.*, vol. 26, no. 3, pp. 11–22, 2002.

[34] G. Bauer, F. Gerstenbrand, and E. Rumpl, "Varieties of the locked-in syndrome," *Journal of Neurology*, vol. 221, no. 2, pp. 77–91, Aug. 1979.

[35] A. Hornof and L. Sato, "EyeMusic: Making Music with the Eyes," in *Proceedings of the 4th International Conference on New Interfaces for Musical Expression*, 2004, p. 4.

SMC-22 Paper Session 7

DESIGN OF TIMBRE WITH CELLULAR AUTOMATA AND *B*-SPLINE INTERPOLATION

Matthew Klassen
DigiPen Institute of Technology, Redmond, WA
mattjklassen@gmail.com

Paul Lanthier
Institut polytechnique Unilasalle
Mont Saint Aignan, France
planthier76@outlook.fr

ABSTRACT

The origin of this paper comes from the collaboration of the authors on the UPISketch software project (see [1]), which is a creation of the Iannis Xenakis Center and a descendant of the UPIC project of Xenakis. The software, created in 2017, allows the user to draw curves which can be interpreted as musical gestures, acting on elements such as pitch, time, and timbre. Two fundamental mathematical tools used in this process are splines, to model graphical gestures, and cellular automata, to generate musical gestures such as pitch sequences. In this paper we apply both of these techniques on the "micro-level" to the *timbre* of waveforms, using the approach of *cycle interpolation*. A waveform is modeled as a sequence of cycles, and each cycle is modeled as a cubic spline curve. The *B*-spline coefficients of each cycle form a discrete representation, which can be manipulated through the use of cellular automata. In this way, key cycles can be generated, and can be interpolated with *B*-splines to form new and interesting timbres. We illustrate this generation and design of waveforms in our own software implementation with JUCE, which in turn will become part of UPISketch.

1. UPISKETCH AND OUR MOTIVATION

In the UPISketch software, sound events are considered as sequences of musical elements such as: time duration, loudness or amplitude, pitch or fundamental frequency, and timbre or waveform. The space of configurations is thus a set of sequences over an alphabet inspired by those parameters. It is straight-forward to give numerical values to the first two parameters, duration and loudness. The pitch can also be represented in discrete segments, such as note values where the pitch is constant, or as continuous curves representing a glissando or other pitch variation. The notion of timbre, however, is recognized as being much more complicated. Perception of timbre can be modeled as multidimensional (see [2] and [3]), and its analysis can be approached in both time and frequency. In [2] the global time-envelope as well as spectral parameters such as spectral centroid are described in detail. We will consider some of these when we discuss our models involving splines and cellular automata. One approach to timbre, employed in many music software systems, is to assign a discrete collection of instrument sounds as available timbres. Another approach is to use synthesis methods, such as frequency modulation, to allow timbres to change continuously according to various parameters.

Our approach to timbre is both discrete and continuous. We model approximately periodic waveforms in terms of cycles, where each cycle is given a discrete representation as a set of cubic *B*-spline coefficients. If a sound is to have a timbre which is somewhat similar to an acoustic instrument, then this should be achievable by a gradual change in time of the shape of cycles, as can be observed in digital instrument sample recordings. A sequence of cycles can be extracted from a recorded sound, or can be generated by artificial means. The method of generation we consider in this paper uses cellular automata, allowing for discrete local changes in the *B*-spline basis coefficients which in turn can be smoothed, or interpolated, to create more gradual changes.

Copyright: © 2022 Matthew Klassen et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. SPLINE INTERPOLATION

It is worth noting that the choice of modeling signals with cubic splines, or piecewise polynomials, has a basic effect on the timbre due to their inherent spectrum. In fact, interpolation with cubic splines can be thought of as a type of additive synthesis of periodic waveforms which are derived from the square wave. It is well known that the square wave has Fourier series with harmonic spectral decay $1/n$, and its integral the triangle wave, has spectral decay $1/n^2$. (See for example [4] chapter 7.) Repeating this process we obtain quadratic and cubic periodic waves, with spectral decays $1/n^3$ and $1/n^4$. Approximation of an audio signal with interpolating cubic splines inherits this type of spectrum, in addition to the modeling of lower frequencies, such as the fundamental, through the shaping of cycles.

We model short samples of an audio waveform at the level of one cycle, or period, although we do not impose the condition that our waveforms are strictly periodic. Further, we choose to represent each cycle as a C^2 interpolating cubic spline on the interval $[0,1]$ given in terms of a *B*-spline basis. Each cycle will be represented by a spline function $f(t)$ with values in the interval $[-1,1]$. We will also assume that the interval $[0,1]$ is evenly partitioned into k subintervals and that f is a cubic polynomial on each subintervals. For simplicity, each cycle will be assumed to

start and end with value 0. It is then natural to use the knot sequence

$$\mathbf{t} = \{t_0, \ldots, t_N\} = \{0,0,0,0,\frac{1}{k},\frac{2}{k},\ldots,\frac{k-1}{k},1,1,1,1\}$$

with associated B-spline basis:

$$\{\mathscr{B}_0(t),\mathscr{B}_1(t),\ldots,\mathscr{B}_{n-1}(t)\}.$$

Each cubic B-spline basis function $\mathscr{B}_i(t)$ uses 5 consecutive knot values: $t_i,\ldots,t_{i+4}$ and is positive on the interval of its support: (t_i,t_{i+4}). The dimension is $n = k+3$ and thus $N = k+6$. In this way, the spline

$$f(t) = c_0\mathscr{B}_0(t) + \cdots + c_{n-1}\mathscr{B}_{n-1}(t)$$

will satisfy $f(0) = 0 = c_0$ and $f(1) = 0 = c_{n-1}$. The remaining B-spline coefficients can be considered a discrete representation of the cycle which can evolve through various discrete transformations. In some cases, cycle data may be derived from actual recorded sounds, or audio files, or cycles may be generated through algorithmic processes, or synthesis. The spline function f can be evaluated by nested linear interpolation, or the de Boor Algorithm:

de Boor Algorithm for B-spline evaluation:

First, set $c_i^0 = c_i$ for $i = 0,\ldots,n-1$. Next for $t \in [0,1]$ choose J so that $t \in [t_J,t_{J+1})$. Then for $p = 1,2,3$, for $i = J-3+p,\ldots,J$, set:

$$c_i^p = \frac{t-t_i}{t_{i+3-(p-1)}-t_i}c_i^{p-1} + \frac{t_{i+3-(p-1)}-t}{t_{i+3-(p-1)}-t_i}c_{i-1}^{p-1}$$

Finally, the output is $f(t) = c_J^3$.

The use of B-splines for audio waveform synthesis occurs in the work of Nick Collins [5], in which one cycle of the waveform can be manipulated by the movement of control points, as well as in the modeling of envelopes for F_0 synthesis as in [6]. Collins' software implementation provides the user with the ability to modulate the timbre of the resulting waveform by moving control points. Our approach is to replace the control by the user of individual cycles with cellular automata for the generation of key cycles. These key cycles are then interpolated to create "in between" cycles with B-spline interpolation. This extends the work in [7], where all key cycles are assumed to come from an audio signal. As initial input to the CA, we may also use one cycle from an original audio sample.

Since we may derive a cycle from actual audio data, we state a few more parameters which determine the model. It is useful to keep the number of subintervals k constant throughout a generated sequence of cycles, and to keep the subintervals of $[0,1]$ of uniform length. Then the dimension of the vector space V of C^2 cubic splines on the sequence of subintervals is $n = k+3$, with basis given as above. In order to approximate one cycle of a recorded audio waveform, we need to have n inputs in the interval $[0,1]$, which are evenly spaced, according to the Schoenberg Whitney Theorem on spline interpolation (see [8]). So we use the $k-1$ knot values which occur strictly inside the interval $[0,1]$, as well as two more values at the midpoints of the outer two subintervals: $\frac{1}{2k}$ and $1-\frac{1}{2k}$. These $k+1$ inputs, together with the end conditions $f(0) = 0 = f(1)$ will then guarantee a unique solution in V. In figure 1 is an example of interpolating cubic spline to one cycle of audio data with 330 sample points. The spline, drawn here in blue, uses $n = 60$ interpolation points.

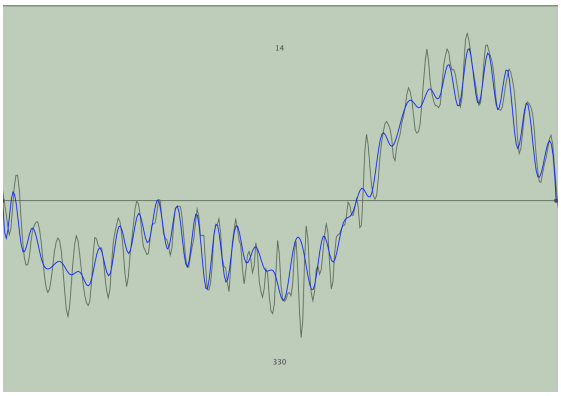


Figure 1. Spline interpolation of audio data

An analogy for this type of sound model comes from animation with key framing. Here, the cycle plays the role of one frame in animation, where the key frames might come from an artist's imaginative drawing, or they might come from motion capture data. In the same way, one key cycle might come from some idea for the design of timbre, or it might come from one cycle which is captured from a particular audio recording. In the previous paper [7] we explore mostly the latter case, but here we explore the former case: the design of timbre.

3. CYCLE INTERPOLATION (CI)

As in [7], we use a second set of splines to control the evolution of the cycles between key cycles. We call this set of splines *meta-splines* and the set of splines which model individual cycles *cycle-splines*. It is reasonable to use the same types for both sets, for instance C^2 cubic splines. However, for computational speed, especially when there are many key cycles to interpolate, we may also use piecewise linear meta-splines.

An important point, raised by Collins in [5], is that linear interpolation between spline coefficients can be equivalent to an "audio crossfade", meaning that the final samples produced are nothing more than the fade out of one block of samples while another is faded in. This is a consequence of the simple observation that a linear combination of B-spline coefficients produces an equivalent linear combination of output values of the spline functions. One way to avoid this type of crossfade is to supply cycles with an envelope, which can be as simple as a sequence of scaling factors for each cycle. Such scaling factors can be derived from realistic envelopes of actual recorded sounds, or generated in other ways. Any nonlinear choice of such scale

factors will avoid the case that the resulting cycle interpolation becomes an audio crossfade.

Our synthesis model consists of a sequence of cycles C_j, some of which are key cycles and are generated by some algorithmic process, and the rest of which are interpolated. There are two main types of key cycle sequences: regular and irregular. The regular case places key cycles at regular time intervals, with a fixed number of intermediate cycles. The irregular case usually places more key cycles near the beginning of an audio sequence or waveform, in order to simulate an instrument sample which begins with an attack and decay phase. Such irregular sequences of positive integers can be generated in various ways, for instance with the classical Fibonacci sequence, or squares, etc.

4. CELLULAR AUTOMATA (CA)

A natural process in music composition is to develop interesting material from a simple element, such as the subject for a fugue, or a melody or phrase. The idea is to generate new elements through musical transformations, such as inversion and transposition. This can be extended further by using other properties, or musical dimensions, such as rhythm, pitch, dynamics, and also timbre. Such transformations of a musical element can be done locally, making changes to a note based on its neighbors for instance. This leads naturally to Cellular Automata (CA), which are computed by a local rule.

CA are already present in the UPISketch software, acting on parameters of pitch and time, as explained in [9], so it is compelling to extend this action to timbre. CA have also been used to model timbre in various ways. For example, in the work of Miranda ([10], [11]) 2-dimensional CA and granular synthesis are used to generate timbres which follow some of the natural properties of acoustically generated sounds. Miranda says: "the cellular automaton models the way in which most natural sounds produced by acoustic instruments evolve: they tend to converge from a wide distribution of their partials to form oscillatory patterns." The grains which are used as "building blocks" are typically $30-40$ ms in length. In our work we also use small segments, or "cycles", about $1-20$ ms of digital audio, represented with a polynomial spline model. The timbre evolves through a sequence of cycles which can be generated with CA, and which can also progress from chaotic spectrum (attack phase) to narrower band oscillatory patterns. We focus here on 1-dimensional CA.

We begin our description of CA as a *local rule* which generates a sequence of n-dimensional vectors

$$\mathbf{v}_k = (c_0, c_1, \ldots, c_{n-1}).$$

This means that we generate vectors iteratively according a function F so that

$$\mathbf{v}_{k+1} = F(\mathbf{v}_k)$$

in some neighborhood $\mathscr{N}_i$ of each coordinate c_i, for $i = 0, \ldots, n-1$. The neighborhood $\mathscr{N}$ specifies a set of indices k containing i. For our application to audio cycles, we suppose that each such vector consists of the B-spline coefficients for one cycle. In order to render audio samples from one such cycle, we evaluate the linear combination of B-splines, the spline function

$$f(t) = c_0\mathscr{B}_0(t) + \cdots + c_{n-1}\mathscr{B}_{n-1}(t)$$

at each of the sample input values t. A typical case could be, for instance, a cycle consisting of 100 output samples generated from a function $f(t)$ with $n = 20$ cubic B-spline coefficients, at sample rate 48 kHz, producing a tone with fundamental frequency 480 Hz.

A simple type of local rule used to define a CA is to average neighboring values. For example, if we denote the coordinates of $\mathbf{x}_{k+1}$ as c'_i:

$$c'_i = \frac{1}{2}(c_i + c_{i-1}).$$

This local rule also can be interpreted as a filter. (To evaluate for $i = 0$ we assume $c_{-1} = 0$.) If we apply this local rule directly to samples, we have a familiar low-pass filter (transfer function $\mathscr{H}(z) = \frac{1}{2}(1 + z^{-1})$), but if we apply it to the B-spline coefficients of one cycle the result is slightly different. The result can be seen through the process of B-spline evaluation with the de Boor algorithm. With degree $d = 3$, each output sample is the result of nested linear interpolation starting from 4 B-spline coefficients. A delay by one B-spline coefficient then has the effect of a delay by one subinterval on the time axis, but only within one cycle. If the subintervals are evenly spaced, this will in turn have the effect of a delay by α samples within the cycle, where α is the number of samples per subinterval, which is not necessarily an integer. If α is in fact an integer, then this filter is similar to an inverse comb filter, with magnitude frequency response having notches at $\frac{f_N}{\alpha}(2i+1)$ for integers i and Nyquist frequency f_N. However, since the local rule is applied one cycle at a time, it does not perform exactly as a filter which is typically applied continuously to an audio stream.

Next we consider another familiar class of CA called Elementary Cellular Automata, which are defined and illustrated here [12]. These CA are set up to act on vectors of 0's and 1's, which can be easily adapted to apply to B-spline coefficients. They are locally defined using one cell and its two neighbors. The output is 0 or 1 depending on the binary digits of a positive integer r, with $0 \le r \le 255$. Let r be written as an 8-digit binary expansion: $r_7 r_6 \ldots r_0$ (so that r_7 is the binary coefficient of 2^7) and let the current vector $\mathbf{x}$ be represented as a binary string $x_0 x_1 \ldots x_{n-1}$. In order to process consecutive triples, also assume we have $x_{-1} = x_n = 0$. Then for each cell (or coordinate) x_i we consider the triple $x_{i-1}x_i x_{i+1}$ as the binary expansion of an integer $b(i)$ from 0 to 7. The local rule then assigns the value $r_{b(i)}$ as the output x'_i. Denote this Elementary CA as $ECA(r)$. We illustrate the local rule for $ECA(30)$ in Table 1. The first row gives the 8 possible triples $x_{i-1}x_i x_{i+1}$ and the second row is the output $r_{b(i)}$.

111	110	101	100	011	010	001	000
0	0	0	1	1	1	1	0

Table 1. local rule for ECA(30)

```
                                   1
                                  111
                                 11   1
                                11  1111
                               11   1    1
                              11  1111  111
                             11   1      1  1
                            11  1111   111111
                           11   1    111       1
                          11  1111  11   1    111
                         11   1      1  1111  11   1
                        11  1111   11  1      1  1111
                       11   1    111   11   11 1    1
                      11  1111  11   111  111   11  111
                     11   1      1 111    1   111   1  1
                    11  1111   11  1   1  11111   1111111
                   11   1    111   1111 1      111        1
                  11  1111  11   111      11   11   1      111
                 11   1      1 111   1   11 111  1111   11   1
                11  1111   11  1   111111   1    1    111  1111
               11   1    111   1111       1111  111  11    1    1
```

Table 2. output from ECA(30), $n = 41$

As in [12], we take as initial input sequence to $ECA(r)$, a vector $\mathbf{x}$ of length n odd, with $x_i = 1$ for $i = \frac{n-1}{2}$, and $x_i = 0$ otherwise. In Table 2, we use $n = 41$ and replace 0's with spaces for ease of visualization.

We apply $ECA(r)$ to B-spline coefficients c_i by multiplication with $a + bx_i$ for some choice of values a and b with $a + b = 1$. This allows the generation of key cycles which are derived from a particular fixed cycle C_0, with B-spline coefficients c_i, $i = 0, \ldots, n-1$. If $a = 1$ then there is no change to the cycles. If $b = 1$ then we have the maximal change as the B-spline coefficients are turned on or off according to the value of x_i in the sequence. Another approach is to borrow a sequence of key cycles from a recorded sound and to apply $ECA(r)$ iterates to this sequence. Again, the amount of change to the cycles can be modified by the choice of a and b. In this way, timbre is modified through the action of CA on the B-spline coefficients of cycles. In the next section we see how this notion of effecting change in other musical elements can also be described through the action of various CA.

5. MULTIDIMENSIONAL MUSIC SEQUENCE

In order to describe the action of CA on music sequences, we begin with the general definition of CA. In general, a 1-D **deterministic cellular automaton** is described by a neighbourhood $\mathscr{N} \subset \mathbb{Z}$, a set $\mathbb{A}$ called an alphabet, with $\mathscr{X} = \mathbb{A}^{\mathbb{Z}}$, and a transformation $F : \mathscr{X} \to \mathscr{X}$ defined by its **local function** $f : \mathbb{A}^{\mathscr{N}} \to \mathbb{A}$ by the relation :

$$F(\boldsymbol{x})_i = f\big((x_{i+k})_{k\in\mathscr{N}}\big) = f\big(x_{i+k_1}, ..., x_{i+k_n}\big).$$

For 2-D automata, $\mathbb{Z}$ must be replaced by $\mathbb{Z}^2$, and for 3-D automata by $\mathbb{Z}^3$. A simple but essential CA is called the **shift map** acting the following way : $\forall \boldsymbol{x} \in \mathbb{A}^{\mathbb{Z}} : \sigma(\boldsymbol{x})_i = x_{i+1}$. For every CA F we have that $F \circ \sigma = \sigma \circ F$. We introduce two automata with $\mathscr{N} = \{0, 1\}$, $\tau = \sigma + \mathrm{id}$ and $\Delta = \sigma - \mathrm{id}$. The first one is called the *Ledrappier automaton* and the second one the *Vieru automaton*. Those two CA have been widely studied in [13, 14]. $f_\tau(a, b) = a + b$ and $f_\Delta(a, b) = b - a$.

Next, we define a music sequence as a sequence of sound events, each of which can be described by pitch α, duration β, and timbre γ. We restrict here to these parameters simply to illustrate with a basic example. Each of these elements can be described in an absolute sense, or a relative sense. For pitch, for instance, the absolute sense could be given as fundamental frequency in Hz, or as a letter name such as $A4$ to indicate fundamental frequency restricted to the equal temperament system based on $A4 = 440$ Hz. The relative sense for pitch describes the interval compared to the previous pitch, which could be given by a frequency ratio such as $\frac{3}{2}$, or a musical interval interval name such as Just Perfect Fifth. In the equal temperament system intervals can be simply indicated as an integer number of semitones, or if the sequence of pitches is considered under octave equivalence, as integers modulo 12, or $\mathbb{Z}_{12}$. In a similar way we can express the time values of notes in absolute or relative time (or duration) since the previoius note onset. All of these concepts are of course displayed in traditional music notation. The element of timbre, however, tends to be treated quite differently. In the broadest sense, when one scores for orchestra there is a discrete set of instruments representing different timbres. Also, solo instruments, such as bowed or plucked strings, have timbre markings such as *sul ponticello* or *sul tasto*, indicating notes to be played near the bridge or over the fingerboard with the resulting change in timbre. All of these notions of timbre are typically thought of as absolute, meaning that a note is either played by a flute or violin, or perhaps a note is indicated as *sul ponticello* on a cello. Relative changes in timbre can often be left up to the performer. With the approach of CA applied to our spline models, it is possible to introduce relative changes to timbre both within individual notes and for a sequence of notes.

Next we present an example of a music sequence based on relative measure of pitch, rhythm and timbre. For simplicity, we assume a discrete time axis with basic unit of time given as one eighth note, and we use a binary string to indicate the onset of a note as 1 and the absence of a note, or silence, as 0. Notes are assumed to be in equal temperament, and intervals between notes are referred to by number of semitones. To make the rhythmic pattern a bit more interesting, we introduce the famous Fibonacci word,

$$\boldsymbol{t}_{\mathrm{fib}} = 0100101001001010010100100101001001\ldots$$

which can be defined recursively by concatenation

$$S_i = S_{i-1}S_{i-2}$$

with $S_0 = 0$ and $S_1 = 01$. This Fibonacci word is a Sturmian sequence, which means that it is aperiodic and balanced (having precisely $n+1$ distinct factors, or subwords, of length n for any positive integer n.) Also, let $\mathbf{1}$ be the constant sequence of 1's, so that $\boldsymbol{t}_{\text{fib}} + \mathbf{1}$, with addition mod 2, is the complement to $\boldsymbol{t}_{\text{fib}}$.

Now we define two music sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ by first specifying pitch and rhythm. Suppose that the pitch of $\boldsymbol{x}$ simply alternates between C and E four semitones higher. As a relative pitch sequence we would then have

$$\alpha_{\boldsymbol{x}} = \{4, -4, 4, -4, \ldots\}.$$

Suppose also that the onsets of the notes for $\boldsymbol{x}$ are given by the 1's in the sequence $\boldsymbol{t}_{\text{fib}} + \mathbf{1}$. Then the relative sequence of note durations for $\boldsymbol{x}$, in numbers of eighth notes, is

$$\beta_{\boldsymbol{x}} = \{2, 1, 2, 2, 1, 2, \ldots\}.$$

Now suppose that the pitch of $\boldsymbol{y}$ simply increases by major thirds, starting with $B\flat$, so that the relative pitch sequence is

$$\alpha_{\boldsymbol{y}} = \{4, 4, 4, \ldots\},$$

and that the onsets for the notes in $\boldsymbol{y}$ are given by the sequence $\boldsymbol{t}_{\text{fib}}$. Then the relative sequence of note durations for $\boldsymbol{y}$ is

$$\beta_{\boldsymbol{y}} = \{3, 2, 3, 3, 2, 3, \ldots\}.$$

The sequences $\mathbf{x}$ and $\mathbf{y}$ are realized on the staff in figure 3, allowing for octave equivalence. In this example we can think of $\boldsymbol{x}$ and $\boldsymbol{y}$ as two initial subjects which are then modified using Δ and τ to produce (very simple) countersubjects which enter one measure later in upper part of the staff. Also note: the starting pitches in each voice are somewhat arbitrary, the focus being on relative pitch in the sequences. (See figure 3.)

Next, we need to describe the timbre for $\boldsymbol{x}$ and $\boldsymbol{y}$. Suppose we choose an initial wave form, based on a single cycle C_0, such as the one at the top in figure 4. Such a cycle can be used with a simple envelope to specify one note at a given fundamental frequency or pitch. The simplest case would be to use this one cycle for all the notes in the example, so that the relative change in timbre is zero, or no change. This would mean the timbre sequence

$$\gamma_{\boldsymbol{x}} = \gamma_{\boldsymbol{y}} = \{0, 0, 0, \ldots\}.$$

Can we change the timbre by indicating a single value to replace 0? Typically timbre is much more complicated, but to continue this basic example, we could simply adjust one B-spline coefficient. For example, suppose we adjust c_r for a fixed choice of r, with $0 < r < n-1$, by some small non-zero value. If we continue to do this, making small changes to c_r in the cycle C_0 as we progress through the notes, we will have a sequence of notes with slightly different timbres and also a simple numeric sequence giving the relative changes in timbre. For example, let's suppose that

$$\gamma_{\boldsymbol{x}} = \{.01, .01, .01, \ldots\}$$

and

$$\gamma_{\boldsymbol{y}} = \{-.01, -.01, -.01, \ldots\}.$$

Finally, consider the action of Δ and τ on $\boldsymbol{x}$ and $\boldsymbol{y}$. Each action is applied to α, β and γ, by the local rules:

$$\Delta(\boldsymbol{x})_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i \text{ and } \tau(\boldsymbol{x})_i = \boldsymbol{x}_i + \boldsymbol{x}_{i+1}.$$

When CA are used on various musical elements, the action splits.

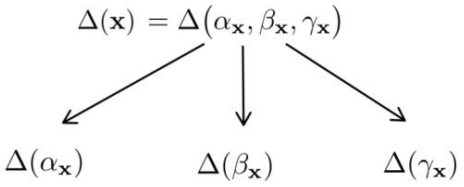


Figure 2. CA action splitting

In the case of pitch, we see that

$$\Delta(\alpha_{\boldsymbol{x}}) = \{-8, 8, -8, 8, \ldots\}, \tau(\alpha_{\boldsymbol{x}}) = \{0, 0, 0, \ldots\},$$

$$\Delta(\alpha_{\boldsymbol{y}}) = \{0, 0, 0, \ldots\}, \text{ and } \tau(\alpha_{\boldsymbol{y}}) = \{8, 8, 8, \ldots\}.$$

In the case of rhythm, we apply the rule directly to the binary sequence and use addition mod 2, which gives the same result for all four:

$$\Delta(\beta_{\boldsymbol{x}}) = \Delta(\beta_{\boldsymbol{y}}) = \tau(\beta_{\boldsymbol{x}}) = \tau(\beta_{\boldsymbol{y}})$$

$$= \{1, 1, 0, 1, 1, 1, 1, 0, \ldots\}.$$

In the case of timbre, we have:

$$\Delta(\gamma_{\boldsymbol{x}}) = \{0, 0, 0, \ldots\} = \Delta(\gamma_{\boldsymbol{y}}),$$

$$\tau(\gamma_{\boldsymbol{x}}) = \{.02, .02, .02, \ldots\} \text{ and}$$

$$\tau(\gamma_{\boldsymbol{y}}) = \{-.02, -.02, -.02, \ldots\}.$$

The above is illustrated in the following very simple multidimensional musical fragment in figure 3.

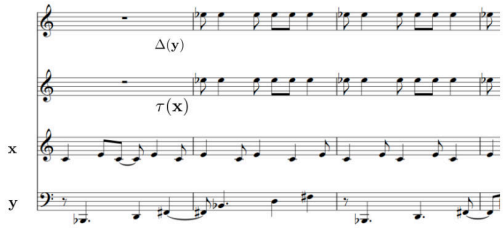



Figure 3. Multidimensional example

6. PROBABILISTIC AND OTHER TYPES OF CA

We saw in the application of $ECA(r)$ to B-spline coefficients that it can be useful to choose constants a and b to form a multiplier other than simply 0 or 1 to produce changes. This approach can be taken a step further by

introducing random variables into the CA computations, which are called **probabilistic cellular automata**. We introduce here the probabilistic versions of Δ and τ called Δ_ε and τ_ε. The idea is simply to perform the local rule update with probability $1-\varepsilon$. For example, the process of updating the n B-spline coefficients of one cycle now involves a sequence of flips of a biased coin, with probability of Heads being $1-\varepsilon$. For each coefficient, the output will be determined by the (deterministic) CA if the flip is Heads, and otherwise the coefficient is not updated. Hence the number of B-spline coefficients which experience a change in this process is a binomial random variable.

We can also add small perturbations p to the B-spline coefficients to obtain further probabilistic CA's. For example, define $\Delta_{\varepsilon,p}$ to have the same function as Δ_ε, except that when the local rule is applied we are also adding a small value, or perturbation, p to the B-spline coefficient. We can set p to be constant for a sequence of cycles generated from this CA, or we could also replace p by X_p, a random variable with uniform distribtion on the interval $[-p,p]$. Thus the CA Δ_{ε,X_p} would have local rule applied to any B-spline coefficient with probability $1-\varepsilon$ give as:

$$c_i' = c_i - c_{i-1} + X_p$$

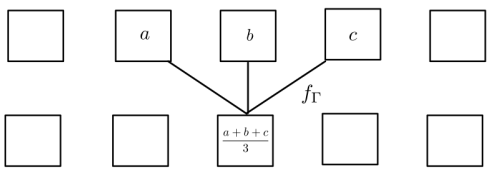
where $X_p \in [-p,p]$.

In figure 4 we show the comparison of Δ and Δ_ε applied to a cycle taken from a guitar pluck recording. Here we use the value $\varepsilon = 0.2$.

We can also define the automata Γ, similar to τ by

$$\Gamma(\boldsymbol{x})_i = f_\Gamma(x_{i-1},x_i,x_{i+1}) := \frac{x_{i-1}+x_i+x_{i+1}}{3}.$$

We describe its action on cells :



Again, Γ behaves as an inverse comb filter on the signal values generated by the spline function, on the portion of one cycle which has uniform knot sequence. We illustrate the differences between Γ and Γ_{ε,X_p} on the same initial cycle in figure 5, with values $\varepsilon = 0.2$ and $p = 0.3$.

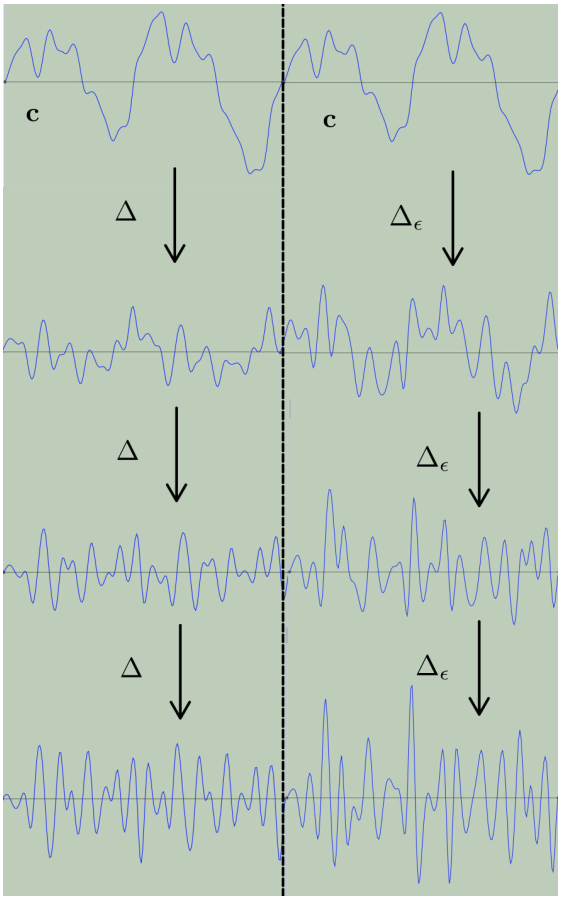


Figure 4. One cycle evolving through Δ and Δ_ε

6..1 Cycle Interpolation (CI) Types

A nice model of synthesized sound can be developed from these ideas. We begin with one cycle C_0, perhaps taken from a recorded sound or generated with sinusoids, or a particular selection of initial B-spline coefficients. Such a cycle can be thought of on the level of one grain, in comparison to granular synthesis. Next, specify an envelope, which can also be borrowed from an actual recorded sound, or can be formed as ADSR or other methods. The waveform is then generated from these intial data by the use of CA and CI. This assumes a choice of the placement of key cycles (regular or otherwise) with all intermediate cycles to be generated by CI. Finally, the CI process also involves the choice of meta-splines, cubic or linear for instance. If the CA is τ, then the iteration of cycles $C_j = \tau(C_{j-1})$. Thus, the gradual evolution of timbre is formed by both CA and CI. We will refer to the sequence of cycles generated by the above method as *Type I*. As a special case, one could also apply this method to every cycle of a sequence, with no CI. In figure 6 cycles 0, 5 and 10 are key cycles, and the rest use CI with piecewise linear meta-splines. Cycle 0 is taken from an audio sample of a guitar pluck at roughly A440, and cycles 5 and 10 are iterates of this cycle using the automata Γ_ε defined above.

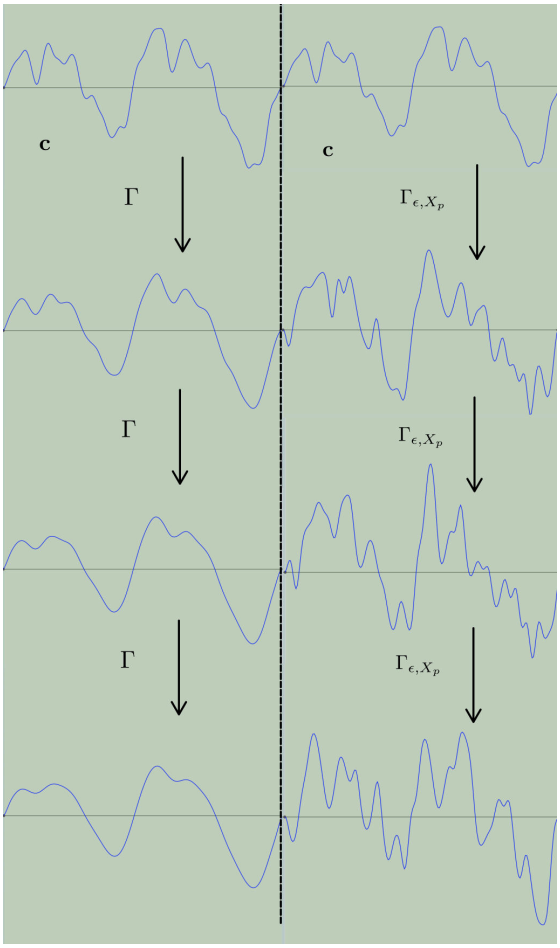


Figure 5. One cycle evolving through Γ and Γ_{ε,X_p}

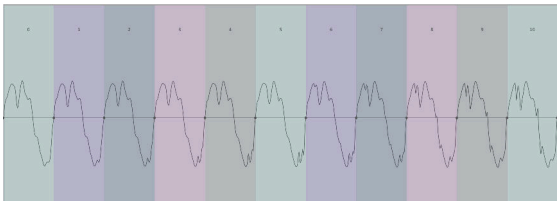


Figure 6. Type I CA with CI example

6..2 Higher scale action

In the above we have applied the CA as a local rule acting on one cycle's *B*-spline coefficients. We can also instead apply this local rule across cycles but focusing on one *B*-spline coefficient. For instance, the rule to change the i^{th} *B*-spline coefficient c_i^j of cycle C_j could use the triple of coefficients

$$c_i^{j-1},\ c_i^j,\ c_i^{j+1}.$$

Doing this for all $i = 0,\ldots,n-1$, and $j = 1,\ldots,n-2$ we can modify a sequence of cycles. This can be applied either to the key cycles, or to an entire sequence of cycles.

This can be interesting to apply to an initial sequence which was not of Type I. This allows the set of key cycles K now to be generated by any process, whether they are taken from various audio samples, or specific basis choices, or random. Call this intial sequence K_0. Then we can iterate the CA to form new sets of key cycles K_r, and in turn generate new sequences of cycles by CI. A sound sample generated in this way will be called *Type II*. As with Type I, we also have the special case where there is no CI, or equivalently every cycle is key.

7. CONCLUSIONS

We saw that this work takes its origin from the concrete context of UPISketch. The CA were already used on pitches and rhythms to generate new material, starting from an initial configuration. We presented here an approach to generate audio signals with evolution of cycles. The method of combining cellular automata and cycle interpolation is capable of modeling timbres which are close to realistic sounds, such as musical instruments, and also designing new and interesting timbres. We have an implementation, using JUCE, to illustrate both the graphical and audio examples discussed in this paper. We are also motivated to contribute convincing audio examples in the UPISketch software, showing the design techniques presented in this paper. Further, this work continues the goal of exhibiting the multi-dimensional action of the CA on musical elements.

8. References

[1] Bourotte, R., Kanach, S., *UPISketch: The UPIC idea and its current applications for initiating new audiences to music*. Organised Sound, vol 24, no.3, 252-260, 2019.

[2] Hajda, J., M., *The Effect of Dynamic Acoustical Features on Musical Timbre*. Analysis, Synthesis, and Perception of Musical Sounds, J.W. Beauchamp, editor, Springer, 2013, ISBN 10: 0-387-32496-8, pp 250-271.

[3] Donnadieu, S., *Mental Representation of the Timbre of Complex Sounds*. Analysis, Synthesis, and Perception of Musical Sounds, J.W. Beauchamp, editor, Springer, 2013, ISBN 10: 0-387-32496-8, pp 272-319.

[4] Steiglitz, K.: *A Digital Signal Processing Primer, with Applications to Digital Audio and Computer Music*, Addison-Wesley, (1996).

[5] Collins, N.: *SplineSynth: An Interface to Low-Level Digital Audio*. Proceedings of the Diderot Forum on Mathematics and Music, Vienna, 1999, ISBN 3-85403-133-5, pp 49-61.

[6] Ardaillon, L., Degottex, G., Roebel, A.: *A multi-layer F0 model for singing voice synthesis using a B-spline representation with intuitive controls*. Interspeech 2015, Sep 2015, Dresden, Germany. hal-01251898v2.

[7] Klassen, M.: *Spline Modeling of Audio Signals and Cycle Interpolation*, Mathematics and Computation in Music, MCM 2022, https://azrael.digipen.edu/research/

[8] de Boor, C.: *A Practical Guide to Splines*, revised edition, Springer-Verlag, New York (1980)

[9] Lanthier, P.: *Multi dimensionnal action of cellular automata on music sequences*, preprint, 2022. https://azrael.digipen.edu/research/

[10] Miranda, E.R. (2003) *Evolving Cellular Automata Music: From Sound Synthesis to Composition*, https://www.researchgate.net/publication/228862474

[11] Miranda, Eduardo R., and Alexis Kirke. "Game of life music." Game of Life Cellular Automata. Springer, London, 2010. 489-501.

[12] Weisstein, Eric W. "Elementary Cellular Automaton." From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/ElementaryCellularAutomaton.html

[13] Lanthier, Paul and Guichaoua, Corentin and Andreatta, Moreno (2019) *Reinterpreting and Extending Anatol Vieru's Periodic Sequences Through the Cellular Automata Formalisms*, Mathematics and Computation in Music, Springer International Publishing.

[14] Lanthier, Paul (2020), *Aspects ergodiques et algébriques des automates cellulaires*, Université de Rouen, PhD thesis.

Visualization of Deep Audio Embeddings for Music Exploration and Rediscovery

Philip Tovstogan **Xavier Serra** **Dmitry Bogdanov**
Music Technology Group, Universitat Pompeu Fabra
first.last@upf.edu

ABSTRACT

User interfaces for music exploration and discovery have always been an exciting application of music information retrieval (MIR) throughout the years. However, while discovering new music is a common goal of such systems, there has been less attention paid to the exploration and rediscovery within personal music collections, where finding interesting relations between music items already familiar to the user can lead to a different type of highly engaging and rewarding experience. In this paper, we present a novel web interface to visualize music collections using the audio embeddings extracted from music tracks. The system allows exploring the relationship between music tracks from multiple perspectives, displaying embedding and tag spaces extracted by music auto-tagging models, trained using different architectures and datasets, coupled with various 2D projection algorithms. We conduct a user study to analyze the effectiveness of different visualization strategies on the participants' personal music collections, particularly for playlist creation and music library navigation and rediscovery. Our results show that such an interface provides a good alternative to standard hierarchical library organization by metadata.

1. INTRODUCTION

In the age of prevalent digital streaming, exploration and discovery of the music are usually done either by the discovery playlists (generated algorithmically or curated) or the users actively looking for new music by themselves. However, the paradigm of music discovery in streaming services neglects the listeners who might want to re-engage with their personal music collections, gathered, curated, and appreciated by their maintainers throughout the years [1]. For such users, exploring their own curated music selections can be a pleasurable and rewarding experience, helping to appreciate and re-contextualize relations between music items and rediscover artists or tracks that they haven't listened to in a long time.

It can be especially relevant in the context of digital music downloads, which still have a considerable impact on independent music distribution [2] (e.g., Bandcamp [1] has gained growing digital sales over the past years with a strong following among music enthusiasts). In this context, many music consumers, and also musicians, DJs, radio hosts, music journalists, archivists, and other professionals or hobbyists that work with digital music collections can benefit from exploration and rediscovery functionality.

Research on user search behavior in the context of music streaming services identified two mindsets: *focused* and *non-focused* [3]. In the focused mindset, users know what they are looking for; and in non-focused, they only have a rough idea. While it was studied in the context of the complete catalog of the music available on the streaming services, those mindsets also apply to the users that mostly listen to their personal collection. The situation that we want to address in this paper can be summarized in the following sentence: *The user doesn't know what he wants to listen to (non-focused mindset) but wants to listen to something familiar (from personal collection).*

The interfaces for music exploration and discovery are quite homogeneous in the industry. The recommendations are usually presented in the form of the playlists or artists, and if the user wants to browse their personal collections, the interface follows the hierarchical approach: artist - album - tracks, or playlist - tracks. In the above-mentioned situation, users usually resort to browsing the hierarchical metadata (artists, genres, tags) or playlists to encounter music to listen to. Algorithmically generated playlists from the tracks from the user's library suit the situation in question to a degree, but they don't provide much interaction.

Thus we propose an alternative content-based approach to represent the music in the multidimensional space which can be projected onto the 2D plane for users to see the entire collection at once. The users can interact with it and listen to short excerpts of music that can enable exploration and rediscovery of the forgotten parts of the music library.

With the wide usage of deep learning in music information retrieval, feature extraction moved from careful engineering towards learned features. There are multiple pre-trained feature-extractor models available [4, 5] that can be used to extract embeddings from audio. Often, these embeddings are used as input for dense neural networks for particular downstream tasks [6]. However, they can also be used as a representation of the music within the embedding space.

[1] bandcamp.com

Copyright: © 2022 Philip Tovstogan et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

In this paper, we take advantage of the auto-tagging systems that are trained to predict the music tags (genre, moods, instruments, etc), and use the extracted embeddings and tag predictions to visualize personal music collections. We introduce the interface that allows users to visualize the entire collection or different subsets of their collection in terms of embeddings extracted from different models and compare them qualitatively. We evaluate the interface in terms of how useful it is for the users to explore their library and create a playlist of the music that they have forgotten and would like to rediscover. In addition, we evaluate different models in terms of the users' preferences of the visualizations that have been produced.

2. RELATED WORK

Many research works perform the visualization of the music in 2D space for exploration, navigation, and recommendation [7]. In this section, we will introduce some selected works that present various interfaces.

One of the earliest works is *GenreSpace* [8] that visualizes tracks in 3D space with colors representing genres. More famous interface *Islands of Music* [9] uses a self-organizing map [10] (SOM) for visualizing music as an artificial landscape of the islands (dense clusters) in the ocean (sparse regions). The emerging islands roughly correspond to the genres of music, and the evaluation is performed mostly qualitatively by authors. The extension of the work [11] introduces several views (based on timbre, rhythm, metadata features) and the ability to switch between them. There was also another related work [12] that proposed playlist generation by drawing the trajectory on the map.

In the following years, multiple studies were also using SOM or some variation of it. *NepTune* [13] visualizes the space as a terrain that can be navigated in 3D by a user. The interface was exhibited in public, where the users could explore their collections. *Globe of Music* [14] projects the space onto sphere instead a plane with the use of GeoSOM [15]. *MusicMiner* [16] uses emerging SOM (ESOM) and U-Map to visualize transitions between genre-based groups. *SongExplorer* [17] is a tangible tabletop interface that presents the songs in a hexagonal grid also using SOM to project 7-dimensional emotion feature space to 2D.

Some interfaces used the metadata in various creative ways for visualization, like [18] that focuses on visualizing personal music collections in form of a disc, rectangle, or tree-map organized according to metadata (genre, year) and highlighted according to personal preferences or playlists.

Since 2010 and the emergence of music streaming, the studies started to focus more on web audio and digital collections. A probabilistic projection of personal music collections based on moods [19] is a remarkable study that focused a lot on user evaluation. It uses the mood features that were extracted via the commercial service from personal Spotify libraries. The features are projected with t-SNE [20] and the interface includes background highlighting based on the probabilistic models to show moods with different colors. The system enables playlist generation via both region selection and drawing trajectories. The authors performed a user study with eight participants over the course of two weeks with overall positive responses and multiple useful insights that include preference of region selection over trajectory drawing. The concept of rediscovery has also been mentioned by authors in this work.

MoodPlay [21] is a remarkable 2D interface that visualizes artists on a mood space. While the free-form exploration is supported, the system is presented as a recommendation system with primary functionality in recommending the artists based on moods. The authors conducted a very extensive user study from the perspective of human-computer interaction (HCI) that provides multiple insights. Online implementation of the interface is available [2] .

One common thing in all these works is that the visualization unit is either a music track, artist, or album. As music similarity is a well-researched area of MIR, and it was a task in MIREX until 2014, the similarity on the level of tracks can go only so far until the subjectivity gets in the way [22]. Our approach is to work with the segments of the music tracks on a smaller scale, which might alleviate the subjectivity of the similarity.

Moreover, only several of the mentioned interfaces [13, 18, 19] work with the personal music collections. Our study focuses on the rediscovery of personal music collections and works with the audio files directly without using external commercial services. Also, most of the works, save for a few exceptions [16, 21, 23] have never been released publicly, as they have been used for study as prototypes. Furthermore, there is a lack of music exploration systems using the latest state-of-the-art in MIR, particularly deep embeddings.

Another common issue with the related work is that many of the mentioned papers (except a few notable exceptions) don't perform conclusive user evaluation, which is important for user-centric MIR systems [24]. We conduct a user study to evaluate our system in form of semi-structured user interviews to get feedback and analyze the functionality.

3. MODELS AND IMPLEMENTATION

We use Essentia [3] library [25] to process audio and extract representations. We use the audio embeddings extracted with modern deep auto-tagging models to represent music in the embedding space and distances between embeddings as a measure of similarity (which can be used for the music recommendation [26]).

We use two common auto-tagging architectures that have been pre-trained on two different datasets that are available in Essentia library [6]:

- *MusiCNN* [27] is a CNN with vertical and horizontal convolutional filter shapes motivated by the music domain. It contains 6 layers and 787 000 trainable parameters.
- *VGG* is an architecture from computer vision [28] based on a deep stack of 3×3 convolutional filters

[2] moodplay.pythonanywhere.com
[3] essentia.upf.edu

that had been adapted for audio [29]. It contains 5 layers and 605 000 trainable parameters.

Both architectures have been trained on two datasets:

- *Million Song Dataset* (MSD) [30] — 500 000+ tracks, collaborative tags from Last.fm [4] service.
- *MagnaTagATune* (MTAT) [31] — 5 000+ tracks, tags provided by players of the TagATune game.

While MagnaTagATune is significantly smaller and usually training on larger datasets gives higher accuracy on downstream tasks, it is commonly used in auto-tagging research and thus provides a good second option for the system. The top 50 most frequent tags from each dataset were used for training the models.

We use two layers from the models' outputs to generate the visualizations in our system:

- *Taggrams* - the output layer that provides tag activation values. The dimension of this layer is 50 for all our models, as they have been trained on top 50 tags.
- *Embeddings* - the penultimate layer of the model. The dimension of embeddings is 200 for MusiCNN and $2 \times 128 = 256$ for VGG.

We process the audio with the hop size equal to the receptive field of the model (3 seconds), which means no overlapping of the frames. We call the part of the audio of the size of the receptive field that produces one vector of output values a *segment*. Thus, the track is represented by a two-dimensional array with a vertical dimension equal to the extracted layer dimension, and the horizontal (time) dimension equal to the duration of the track divided by the size of the model receptive field.

The system is implemented in Python as a Flask web app. The code is open-source and available on GitHub [5] under GNU Affero General Public License v3.0. In the rest of this section, we will provide details of the data processing pipeline.

First, the audio is indexed in the new local SQL database [6] with the track, artist, album, and genre metadata imported from ID3 tags. Next, the audio is processed with the Essentia library with the output of several layers. The advantage of using Essentia is not only that the models are easy to use out-of-box, but also that if you have a working CUDA installation, it will be used to do TensorFlow inferencing. We extract both the tag activation values (taggrams) as well as the activations from the penultimate layer (embeddings). The taggram and embedding vectors are stacked for every segment of the audio thus resulting in a two-dimensional representation of the track, which is saved as a .npy file.

After the data for all tracks have been extracted, the PCA projection of the embeddings and taggrams is performed. We also compute STD-PCA projection, where each embedding/taggram vertical dimension is first normalized on the whole population to prevent large variation ranges in the activation values to dominate the PCA-projected space. The taggrams and embeddings are then aggregated into one .npy file per model for the efficiency of the retrieval, and the segments are indexed in the database for easy lookup of the associated track.

[4] last.fm

[5] github.com/MTG/music-explore

[6] SQLite via SQLAlchemy

4. INTERFACE

The interface [7] (Figure 1) is split into several sections: music selection, visualization selection and highlighting. The user can select music to visualize either by selecting the tags of interest, or artists. One of the important aspects of the system is that it doesn't average the individual embeddings of the segments of the song. Each segment is of the appropriate length of the input size of the model (3 seconds for both MusiCNN and VGG architectures).

One point on the graph represents one segment, The reduction slider allows to show fewer segments per track to visualize many tracks at once. The number represents the step size when loading the data, so it shows all segments for a value of 1, skips every other segment for the value of 2, skips two for the value of 3, etc.

The highlighting section allows highlighting one or more artists, tags, albums, or tracks in red color on the graph. It is interesting to see the groupings and spread of the particular subset of the collection in the context of the larger selection of music.

We use Plotly [8] library to visualize the embeddings. It is a robust library that works well for our use case. One of its advantages is that it supports multiple programming languages, so it is possible to generate plots in Python and add the interactivity in JavaScript.

The visualization selection controls above the graphs allow a user to select architecture, dataset, layer, and projection to visualize embeddings. The option names have been anonymized during the user study to remove any bias that the participants might have towards any of the options. Each option can be selected individually to facilitate the comparison of the combinations. For example, the user might only change the dataset while keeping all other fields the same to see how the training dataset impacts the embedding space visualization.

The available architectures, datasets, and layers have been described in Section 3. Among the available projections apart from PCA and STD-PCA we provide t-SNE [20] and UMAP [32]. PCA and STD-PCA are computed after the extraction of the embeddings, t-SNE and UMAP are computed dynamically upon user request. So while they are slower initially, there is a caching layer implemented to prevent repeated computation of the projections of the same subset.

To get an impression of how different the embeddings spaces are, Figure 2 shows one of the users' personal music collections that was used for evaluation (with a reduction value of 20). This collection mostly consists of rock and metal music, highlighted in red is the artist *Enigma* which is tagged as *new age*. While it is mostly concentrated in one part of the visualizations, some combinations of architecture/dataset/layer manage to do a better job at clustering it.

[7] We provide a demo video of the system online: bit.ly/3bLeFLp

[8] plotly.com

Figure 1: System interface

There are several features of the system to facilitate interactivity. The user can listen to the music while hovering or by clicking the point on the graph that represents a segment of the track. Moreover, when the label of the segment is displayed on one graph, the same label for the same segment is displayed on another graph (see Figure 1). This enables easy identification of the same segment on both graphs during an interaction. Moreover, the user can select several segments on one graph with the lasso or box selection, and the corresponding segments will also be selected on the second graph. There are more tools available to zoom in and pan the individual graphs to facilitate delving deeper into the exploration of the cluster of interest.

5. EXPERIMENTS

To test the system we invited 8 users that have some sort of personal music collection to participate in user study from authors' colleagues and friends. We conducted individual semi-structured interviews with each participant to gather feedback and assess the usability and viability of the system. While there are a lot of potential uses for the system, we focus on the use case of exploration and rediscovery of the music in the private personal collections. Two main research questions that we wanted to address are: the feasibility of the system for the exploration and rediscovery of the users' music collection and the comparison of visualizations in terms of usefulness and interest to users.

Before the experiment, the participants were asked to select a subset of their private music collection that they wanted to explore. We recommended the participants limit the subset to no more than 1000 tracks, and in practice, we encountered collections of sizes from 400 to 1200. In the remote setup, we communicated with the participants through chat to help with the data extraction and ensured that the system can run normally on the users' machines. Then we conducted a video conferencing call with the participant sharing their screen. In the live setup, we asked participants to bring the music collection on the external storage device and performed data extraction and setup on the authors' machines. From 8 participants, 1 was interviewed remotely, and 7 — in-person. The data extraction took different times depending on the specification of the user machine: from 1 to 4 hours with an average of 1.5 hours. While the system doesn't require GPU for processing, most of the participants were using machines with CUDA installation, which sped up the extraction process drastically.

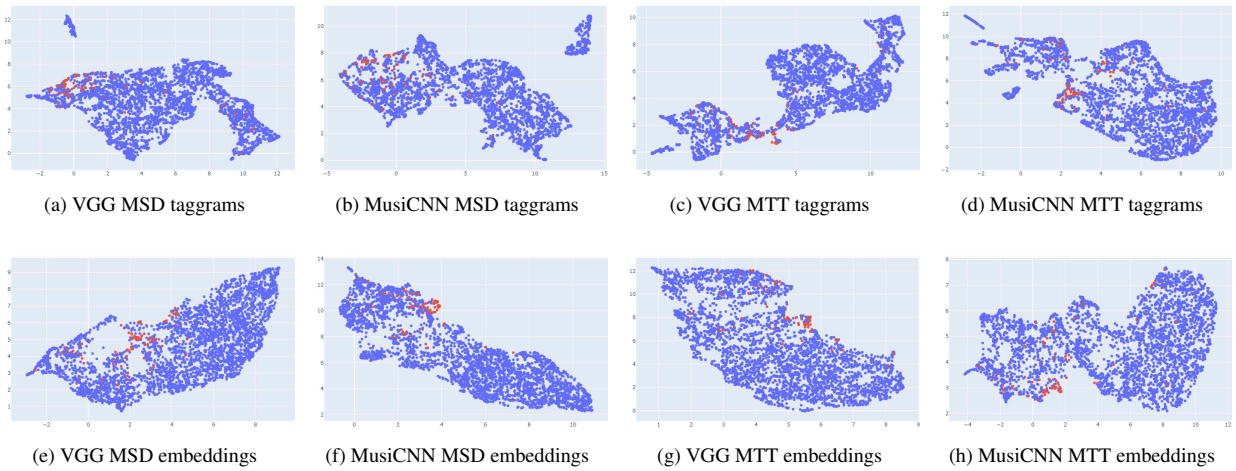
The video and audio from call and audio from the live interview were recorded with the participant's consent for further transcript analysis. The experiment started with the introduction of the features of the system to the participants by reading the introduction text. The text was kept the same to minimize the possible bias. We let participants get familiar, ask questions and play with the system and make sure that they are comfortable with it. The maximum time allocated for the familiarity phase is 10 minutes. We ensured that participants used every control element of the interface at least two times, and if they didn't, we encouraged them to use it. Then we gave the participants a task that was formulated in the following manner: *Imagine that you want to listen to something from your library that you haven't listened to in a while. Explore the system and make a playlist for yourself.*

During the interview, the participants were encouraged to try different settings and engage with the system as much as possible. When they changed the parameters of the visualization (architecture, dataset, layer, and projection), we asked them if they liked or disliked the previous combination. After the users were content with their selection of the tracks for the playlist, we asked them to fill in the questionnaire [9] to assess their thoughts about the system.

The questionnaire is split into two parts: the first part includes background questions such as age, musical training, familiarity with playlists, and experience with listening to music. The authors were present to answer any questions that the users might have about the questionnaire but didn't interfere beyond that.

The second part of the questionnaire contains questions about the system that were designed to identify which fea-

[9] The questions are available online: `bit.ly/3w9xJe0`



(a) VGG MSD taggrams (b) MusiCNN MSD taggrams (c) VGG MTT taggrams (d) MusiCNN MTT taggrams

(e) VGG MSD embeddings (f) MusiCNN MSD embeddings (g) VGG MTT embeddings (h) MusiCNN MTT embeddings

Figure 2: UMAP visualizations of *new age* (in red) in mostly rock and metal collection (reduction of 20)

tures of the system users liked, what did they think about the visualizations on both global and local levels, the usefulness of the system for music exploration, rediscovery and playlist creation. To measure users' opinions and feedback we used the 5-point Likert scale: 1 - Strongly disagree, 2 - Disagree, 3 - Neither agree nor disagree, 4 - Agree, 5 - Strongly agree. Interviewees were asked to be as critical as possible and encouraged to explain their reasoning behind the choices they made as well thinking out loud.

6. RESULTS AND DISCUSSION

The participants of our study are aged 27–39 years with an average age of 30, 7 male and 1 female. All of them have some kind of music training ranging from 1 to 20 years, the median of 6 and an average of 8 years. They listen to music 0.5–8 hours per day with 1 hour or less actively, less than 50% of the time (20% on average) to playlists. The participants create playlists with frequency ranging from every day to rarely with good coverage of all options in between. The frequency of desire to rediscover their music ranges from every day to several times per month, with most of the responses in the latter category. The broad genres covered by the users' personal music collections span mostly electronic, rock and metal.

6.1 Interaction, Exploration and Rediscovery

After analyzing the interviews and the results of the survey (see Table 1) we can see the trend that the system achieves its goal to help users to interact, explore and rediscover personal music collections and create playlists. While the quantitative results are not strong enough due to the small sample size, the focus of this study is on qualitative feedback with every participant having discovered some interesting connections between tracks in their library during the interviews. We performed topic analysis, and the last 2 interviews didn't introduce any new topics, thus providing good support that the most important topics have been covered within 8 participants.

Question	Mean $\pm$ STD
Liked interacting with system	$\mathbf{4.9} \pm 0.4$
Had preference for particular model	$3.6 \pm \mathbf{1.2}$
Preferred over browsing	4.3 ± 0.7
Preferred over random	4.4 ± 0.9
Liked big picture	3.8 ± 1.0
Liked segment groupings	4.4 ± 0.7
Discovered unexpected connections	$\mathbf{4.5} \pm 0.5$
Rediscovered something	$\mathbf{4.6} \pm \mathbf{1.1}$
Want to use for playlist creation	4.1 ± 1.0
Want to use for inspiration	4.3 ± 0.7
Had rewarding experience	$4.1 \pm \mathbf{1.1}$
Had engaging experience	$\mathbf{4.5} \pm 0.8$

Table 1: Summarized results from Likert scale questions

One of the topics that came up in several interviews was about using *segments* instead of tracks, segment length, and possible averaging of the segments. An argument in favor of using segments is that they are short, concise, can represent better the music evolution with time and span multiple tags, and are easier to perceive as a unit. For example, while it might be difficult to say which track is more similar to the reference track, some participants agreed that it is relatively easier to answer the same question with the segments.

However, multiple participants remarked that the length of 3 seconds is too short. While the similarity might be easier to judge, it might not translate well towards track similarity, exploration process and lead towards undesirable behavior during playlist generation. For example, if there is a segment of low-energy music in the cluster of similarly chill tracks, but the segment is an interlude in a much more aggressive track, the track in question will be undesirable in the low-energy playlist. Some participants mentioned that they would prefer segments of at least 10 seconds.

One suggestion that came up multiple times is to *average* embeddings of several segments. It makes sense for the segments that are similar to each other. However, if

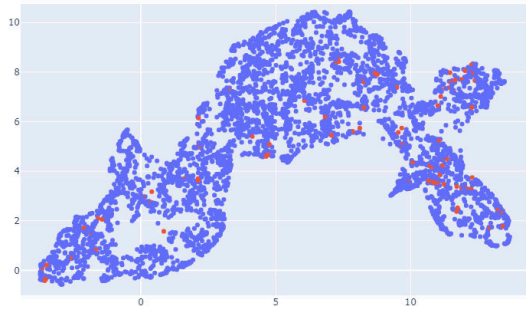


Figure 3: Long complex track highlighted in red

the segments are quite distinct and are from two different regions in the embedding space, taking the average might put the resulting average into a new third region that has nothing to do with the original ones. This problem is exacerbated on a larger scale, where averaging can make tracks that are very complex and span multiple regions in the embedding space (Figure 3) be reduced to several points which are not representative of the dynamics of the track. For certain genres it can be quite a bit problem, for example for different movements in classical music.

Participants' opinions also varied towards the ability of the interface to *visualize the entire collection*. Some of the participants noted that it was nice that all aggressive and high-energy tracks were on one side with the more chill and relaxed tracks on the other side. One participant mentioned *"(pointing at one side of the visualization) here is hard music, music that my mother doesn't like, but if I come here (pointing at the opposite side), it is more peaceful, relaxing"* while moving from one side of the visualization to the opposite one. The semantics gradients that were mentioned as obvious from the big picture are (depending on the architecture/dataset): rock–ambient, electronic–acoustic, vocal–instrumental. Those semantics are indeed represented by the tags from the training datasets, and it is useful to see that the participants agree on those semantics. Few other participants didn't pay any attention to the global distribution and dived right in exploring clusters hovering the mouse over the different regions of embedding space. Some participants enjoyed zooming into random clusters, while others didn't utilize zoom functionality as much.

Rediscovery was the part of the experience that almost all the participants were very happy and vocal about. Ones that weren't particularly keen on rediscovery evaluated the system more in the context of DJing. Encountering artists and tracks that they haven't listened to in a while happened both during the random walks over the entire space and while investigating local clusters. The same can be said about unexpected connections with several participants saying *"I would never think to put these two artists together in a playlist, but it works quite well for these tracks,"* or *"if you listen to segments, they sound quite similar in timbre, what won't happen to full tracks."* Some participants have noted that it was good to have an audio player in the interface because if they would be using the system outside of the interview, they would stop the exploration process and listen to the track that they stumbled upon from start to finish.

Interestingly enough, the *highlighting* functionality of a particular artist / album / track / tag became quite divisive — many participants used it to highlight a tag or an artist either as a seed to go from or as a target that they wanted to explore. It is the functionality that was most often mentioned as a favorite in the questionnaire, however, some participants didn't engage with it after the introduction.

As the tags that the models are trained on are quite generic (guitar, vocal, rock, chill, electronic, etc.), several participants mentioned that the models probably are not capable of distinguishing subtle differences between subgenres of their homogeneous collection by pointing out the segments in some clusters that don't belong together due to the style. One participant noted: *"The similarity is not captured well between different styles of dance music."*

Overall, the participants took between 5 to 10 minutes to get familiar with the system and 2 to 20 minutes to explore it to try different visualizations and make a selection that would produce a playlist that they are satisfied with. However, after they have created the playlist that they were content with, some participants spent a lot of time continuing exploration of other regions of their collection. Several users mentioned that there could be other methods to generate a playlist, for example, track- or artist-based radio that uses the seed segment or track: *"Maybe the system can lasso select tracks for me."* The playlist creation functionality was mentioned multiple times as a very strong use case for using the system after the novelty of interaction would wear off.

6.2 Comparison of Visualizations

Even if the sample size for the comparison study is not large to draw strong conclusions, after analyzing the responses to the question of whether the participants liked or disliked a particular combination of architecture / dataset / layer / projection, some interesting trends can be identified. As mentioned before, all options were anonymized for user testing to remove potential biases. The only option that could be inferred was the projection, as participants could guess which type of the projection it is just by looking at the graphs, however, no participants made it obvious that they recognized any projections.

Several participants mentioned that they liked two visualizations side-by-side and engaged in using both to select subsets. Some participants pointed out that different combinations capture well different aspects of similarity: *"It seems that A2D2 (MusiCNN-MTAT) can separate ambient from drums, while A1D1 (VGG-MSD) clusters the timbral aspect of sounds together well"* and took advantage of that by using both at the same time. The combination VGG-MSD has been mentioned by multiple participants as being good for timbre similarity.

Among architectures, datasets, layers, and projections, participants had the strongest preferences towards projection options. Most participants mentioned that the distribution looks more interesting in UMAP (5) and t-SNE (4) compared to PCA and STD-PCA. We attribute it to both t-SNE and UMAP being non-linear transformations,

and UMAP preserving distances better than t-SNE. Non-linearity helps to represent the local structure better at the cost of the global structure. The common comments in favor of PCA and STD-PCA are that they are faster, and capture the global structure much better. *"P1 (STD-PCA) seems to group the same sounds that I would put together for DJing"*

While participants were encouraged to compare different architectures, datasets, and layers, it took a lot of effort from the participants and was less engaging than the exploration of the visualizations that are already in front of them. We conclude that there should be a separate experiment to present participants with predetermined comparison pairs for proper evaluation. Although, all participants answered positively to the question of them having a favorite combination of architecture / dataset / layer / projection.

Comparing the architectures coupled with datasets, commonly mentioned as good were combinations VGG-MSD (3) and VGG-MTT (3), a bit less MusiCNN-MTT (2). While VGG is an architecture taken from computer vision without many modifications, and MusiCNN takes advantage of the music domain knowledge in the filter design, there was no conclusive evidence for one being preferred more than the other. Taggram layer was mentioned several times in the preferred combinations (4), more than the embedding layer (2). This might indicate that the semantics of the tags is more useful and representative than the deeper layer of the neural network.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we present the interface that allows users to visualize personal music collections. To the best of our knowledge, this is the first study proposing a music exploration interface that uses state-of-the-art deep audio embeddings. Importantly, the system is open-source, the installation process is well documented and it is easily extendable with other models for extracting feature embeddings.

We evaluated our system via semi-structured interviews with the users. From the evaluation results, we can conclude that this interface is engaging and rewarding to use for people when they are in the mood for rediscovery or exploration of personal music collections. Moreover, the results of the questionnaire strongly support the usefulness and viability of the system. We believe that such systems can be extended to the case of music discovery and exploration that is not limited to personal music collections.

While the performed small-scale evaluation provides initial results and insights on the preferences for architectures, training datasets, layers, and projections, a larger study needs to be conducted to gather more data to support our initial findings.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068.

8. REFERENCES

[1] S. J. Cunningham and S. J. Cunningham, "Interacting with personal music collections," in *Information in Contemporary Society (ICS)*. Washington, DC, USA: Springer, Mar. 2019, pp. 526–536.

[2] IFPI, "Global music report 2021," 2021.

[3] A. Li, J. Thom, P. Chandar, C. Hosey, B. S. Thomas, and J. Garcia-Gathright, "Search mindsets: Understanding focused and non-focused information seeking in music search," in *The World Wide Web Conference (WWW)*. San Francisco, CA, USA: ACM, May 2019, pp. 2971–2977.

[4] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA, USA: IEEE, Mar. 2017, pp. 131–135.

[5] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, UK: IEEE, May 2019, pp. 3852–3856.

[6] P. Alonso-Jiménez, D. Bogdanov, J. Pons, and X. Serra, "Tensorflow audio models in Essentia," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Barcelona, Spain: IEEE, May 2020, pp. 266–270.

[7] P. Knees, M. Schedl, and M. Goto, "Intelligent user interfaces for music discovery," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 165–179, Oct. 2020.

[8] G. Tzanetakis, "Automatic musical genre classification of audio signals," in *Proceedings of the 2nd International Symposium on Music Information Retrieval (ISMIR)*. Bloomington, IN, USA: Zenodo, Oct. 2001.

[9] E. Pampalk, A. Rauber, and D. Merkl, "Content-based organization and visualization of music archives," in *Proceedings of the 10th ACM International Conference on Multimedia (MM)*. Juan-les-Pins, France: ACM, Dec. 2002, pp. 570–579.

[10] T. Kohonen, *Self-organizing maps*, 3rd ed., ser. Springer Series in Information Sciences. Berlin, Heidelberg: Springer, 2001, vol. 30.

[11] E. Pampalk, S. Dixon, and G. Widmer, "Exploring music collections by browsing different views," *Computer Music Journal*, vol. 28, no. 2, pp. 49–62, Jun. 2004.

[12] R. Neumayer, M. Dittenbach, and A. Rauber, "PlaySOM and PocketSOMPlayer, alternative interfaces to large music collections," in *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR)*. London, UK: Zenodo, Sep. 2005, pp. 618–623.

[13] P. Knees, M. Schedl, T. Pohle, and G. Widmer, "An innovative three-dimensional user interface for exploring music collections enriched," in *Proceedings of the 14th ACM International Conference on Multimedia (MM)*. Santa Barbara, CA, USA: ACM, Oct. 2006, pp. 17–24.

[14] S. Leitich and M. Topf, "Globe of music - music library visualization using geosom," in *Proceedings of the 8th International Society for Music Information Retrieval Conference (ISMIR)*. Vienna, Austria: Zenodo, Sep. 2007, pp. 167–170.

[15] Y. Wu and M. Takatsuka, "Spherical self-organizing map using efficient indexed geodesic data structure," *Neural Networks*, vol. 19, no. 6-7, pp. 900–910, Jul. 2006.

[16] F. Mörchen, A. Ultsch, M. Nöcker, and C. Stamm, "Databionic visualization of music collections according to perceptual distance," in *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR)*. London, UK: Zenodo, Sep. 2005, pp. 396–403.

[17] C. F. Julià and S. Jordà, "SongExplorer: a tabletop application for exploring large collections of songs," in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*. Kobe, Japan: Zenodo, Oct. 2009, pp. 675–680.

[18] M. Torrens, P. Hertzog, and J. L. Arcos, "Visualizing and exploring personal music libraries," in *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR)*. Barcelona, Spain: Zenodo, Oct. 2004.

[19] B. Vad, D. Boland, J. Williamson, R. Murray-Smith, and P. B. Steffensen, "Design and evaluation of a probabilistic music projection interface," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. Málaga, Spain: Zenodo, Oct. 2015, pp. 134–140.

[20] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, Nov. 2008.

[21] I. Andjelkovic, D. Parra, and J. O'Donovan, "Moodplay: Interactive music recommendation based on artists' mood similarity," *International Journal of Human-Computer Studies*, vol. 121, pp. 142–159, Jan. 2019.

[22] A. Flexer, "On inter-rater agreement in audio music similarity," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan: Zenodo, Oct. 2014, pp. 245–250.

[23] M. Hamasaki, M. Goto, and T. Nakano, "Songrium: Browsing and listening environment for music content creation community," in *Proceedings of the 12th Sound and Music Computing Conference (SMC)*. Maynooth, Ireland: Zenodo, Jul. 2015, pp. 23–30.

[24] M. Schedl and A. Flexer, "Putting the user in the center of music information retrieval," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*. Porto, Portugal: Zenodo, Oct. 2012, pp. 385–390.

[25] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra, "Essentia: An audio analysis library for music information retrieval," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*. Curitiba, Brazil: Zenodo, Nov. 2013, pp. 493–498.

[26] A. Ferraro, X. Favory, K. Drossos, Y. Kim, and D. Bogdanov, "Enriched music representations with multiple cross-modal contrastive learning," *IEEE Signal Processing Letters*, vol. 28, pp. 733–737, Apr. 2021.

[27] J. Pons and X. Serra, "MusiCNN: Pre-trained convolutional neural networks for music audio tagging," Sep. 2019.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations (ICLR)*. San Diego, CA, USA: arXiv, May 2015.

[29] K. Choi, G. Fazekas, and M. B. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*. New York City, NY, USA: Zenodo, Aug. 2016, pp. 805–811.

[30] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*. Miami, FL, USA: Zenodo, Oct. 2011, pp. 591–596.

[31] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: the case of music tagging," in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*. Kobe, Japan: Zenodo, Oct. 2009, pp. 387–392.

[32] L. McInnes, J. Healy, and J. Melville, "UMAP: uniform manifold approximation and projection for dimension reduction," Feb. 2018.

Polyfōnía: an Interactive Vocal Processing System using Finger Tracking

Areti Andreopoulou
Laboratory of Music Acoustics
and Technology (LabMAT),
National and Kapodistrian
University of Athens
a.andreopoulou@music.uoa.gr

Natalia Kotsani
School of Electrical
and Computer Engineering,
National Technical
University of Athens
nkotsani@corelab.ntua.gr

ABSTRACT

Traditional foot-controlled vocal processors are not oriented towards the real-time needs of vocal performers, as their handling method interferes with the posture and expressiveness of the performers' body, acting more like a communication barrier and a source of distraction. Polyfōnía is a real-time portable vocal processing system, which can be controlled by means of finger gestures without the use of gloves or the need for wiring. Its vocal processors include a live looping system and a harmonizer. Motion detection is performed using a motion sensor. Through specific gestures, users can develop and control real-time voice harmonies and live looping. The system takes as input audio signals from a microphone and outputs a processed version of the input. The goal of Polyfōnía is to offer a more natural handling of vocal processors, through new technologies and interactive performance, eliminating the need for added components and making real-time vocal processing a valuable tool for artists. It is intended to be used in conjunction with the vocalists' standard performing techniques and aesthetics, as a means for expanding the creative possibilities of the human voice.

1. INTRODUCTION

Interactive experiences can be classified into different taxonomies, depending on the participants' level of engagement with the artworks [1] or on their reliance or independence of past information [2]. Analysis of the different types of interaction, in combination with a deeper understanding of their changing nature, could lead to the optimization of the interactive music systems, thus providing richer user experiences and a significant expanse of the artistic creation scope. The idea behind the design of Polyfōnía is the development of an interactive system for controlling digital audio effects, that would be consistent with the interpretation of phonetic music, i.e., in complete harmony with both its kinesiological terms and its performance requirements.

Copyright: © 2022 First author et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

According to Luck and Toiviainen [3], the position of a singer's upper body and the inclination of their head have a significant effect on the quality of their voice, while normal postural alignment optimizes their vocal performance, in terms of vocal technique [4–6]. The operation of foot-controlled digital processors poses specific demands (i.e. altering one's body posture and tilting their head), many of which adapt more to the instrumentalists' needs. In music genres where the essence of the vocal performance is borne by the optimization of one's vocal technique, expressive austerity, or theatrical interpretation, controlling processors through pedals, knobs, or even midi keyboards often creates additional needs, and, in the latter case, assumes prior musical training/expertise, rather than embracing the performance in a natural manner.

The key-requirements in the design of interaction parameters in Polyfōnía are:

- operational without the use of cables, gloves, or bulky devices
- tracking of both hands
- use of hand-gestures that are easy to remember and congruent with the performers' normal postural alignment, thus not interfering with their expressive body movement
- operational in low-light conditions
- immediate and efficient system response, not requiring the use of a display

These design requirements lead to a system assisting users control vocal polyphonic creation in real time, even in poor light conditions, with the best accuracy, using the simplest possible equipment.

Although there is extensive literature on voice gesture-controllers, presented in the next section, there is a significant gap in the implementation of voice polyphonic controllers that would not rely on the use of gloves or wiring. The motivation for this research was the possibility of polyphonic interpretation of traditional Greek songs, live, on stage, without the use of additional cabling equipment, so that such a task could be integrated with the aesthetics and the simplicity of this type of music.

The novelty and contribution of this work consists in the following: (i) creating a gesture-controlled system for the

production of voice polyphony without using gloves, joysticks, midi keyboard, or wiring (ii) designing the gestures and functionality that would support the simplicity in the application use and would not require extensive user-training. (iii) ensuring the extendability of the system, allowing users to add operations and enrich its use.

2. RELATED WORK

Motion-capture is a contemporary tool utilized in expressive music interpretation systems [7] since the sophisticated movements used in human-computer interaction make it an efficient method for controlling expressive parameters. Sensors or computer vision algorithms have been used creatively for decades [8–10]. Examples of such creative control systems are listed in this section.

In 2003, Dobrian and Bevilacqua [11] developed a gestural control system for music, which received data from Vicon 8. The system input required the circular placement of eight cameras around the user. Another large-scale system was Pamela Z [12], which uses extended vocal techniques and live, body-controlled electronic processing from solo events to large-scale works that combine video, audio, and live musicians / singers / actors.

Wearable devices have also been used in the context of creative gestural controllers. For example, in 2009 Jessop [13] created a gesture-based wearable controller for live vocal performances, the Vocal Augmentation and Manipulation Prosthesis (VAMP), allowing singers to capture and manipulate single notes. In a similar concept, Gualtieri et al. [14] and Feugère et al. [15] have proposed the PLS and Chorus Digitalis polyphonic gestural singing systems, respectively, which operate using gloves, joysticks, and graphic tablets. In 2018, Tavares et al. [16] introduced the WMTSensor-GloveNorderval, a data-glove used in an opera performance to control audiovisual elements on stage through gestural movements. This glove-like wireless wearable controller can communicate with a multitude of software through Open Sound Control (OSC) messages.

Sound spatialization is another field that has been explored in this context. In 2009, Smallwood et al. [17] developed a hemispherical 6-speaker system that mimics the way acoustic musical instruments emit sound in all directions, which users could control with vertical and horizontal movements. Berthaut et al. [18] designed a virtual multi-process immersive instrument to create audio events in musical environments using 3D audiovisual objects.

In [19], Hantrakul and Kaczmarek mention the $[A]^2$ project, which uses a system based on the Leap Motion Controller that modulates the effects of a live remix using sampled recordings of a preceding performance and a 5-grain granular synthesizer. The system, which is built in Max/MSP, allows users to trigger individual audio grains through finger motion. Abacus is another Max/MSP operated system developed on an Arduino Teensy built by Warren [20]. It tracks vocal gestures and uses them to adjust various musical parameters including rhythm, pitch, and noise.

Norderval et al. have focused on the use of sound technologies as a prosthesis to the vocal body [21] and the acoustic and aesthetic conflicts between opera and electroacoustic music [22]. Their system is based on Max/MSP and a variety of controllers such as the Leap Motion , smart phones, the HotHand wearable ring etc. In a similar concept, Eriksson et al. have studied the movement-based interactions, implementing an opera where custom-built drones are performing on stage alongside human performers [23], analysing how movements enabled by the human-drone assemblage can affect the artistic expressions [24].

3. SYSTEM ARCHITECTURE

Polyfōnía consists of a Max/MSP standalone patch with two main Interaction Modules, each associated with a Vocal Processor, and a Motion Skeletal Tracking component. Its two vocal processors are the Live Looping Module and the Harmonizer Module. Polyfōnía's simple architecture, requires a Leap motion controller, a microphone, and a computer. The audio input from the microphone is routed to the Vocal Processors. The input sensor data collected from the motion controller passes through the Motion Skeletal Tracking component to the main Interaction Modules. The audio signal is modified by the Processors depending on the Interaction Module output. The implementation can be found here [25].

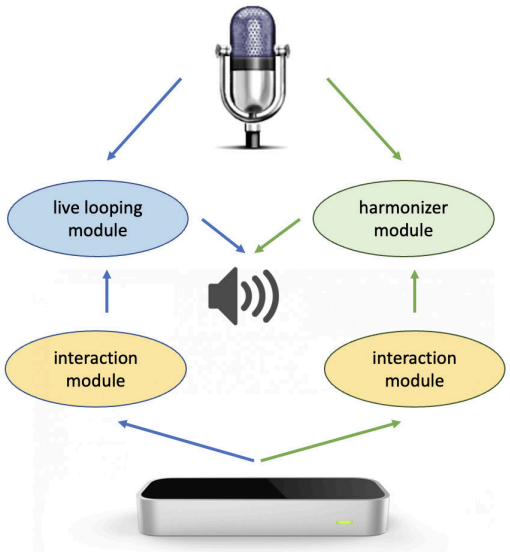


Figure 1. System architecture: The audio signal is the input to the live looping and the harmonizer modules, described in 3.4. The sensor data, extracted from the Leap Motion, is the input of the two interaction modules, described in 3.5. Each of the two interaction modules controls one of the vocal processors. The final processed audio signal is routed to MAX/MSP for output.

3.1 The Motion Controller

The motion detector that was used was the well-known off-the-shelf Leap Motion Controller, designed by Leap Mo-

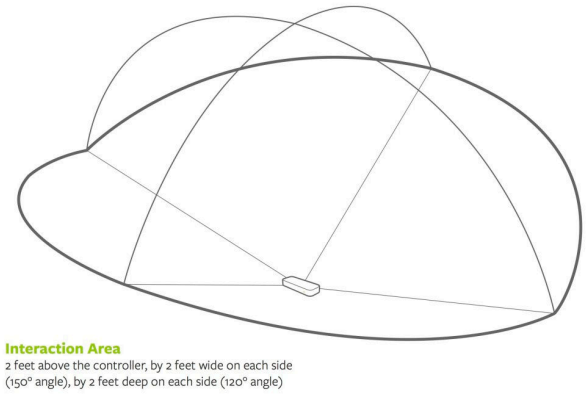


Figure 2. The Leap Motion Interaction Area [29]

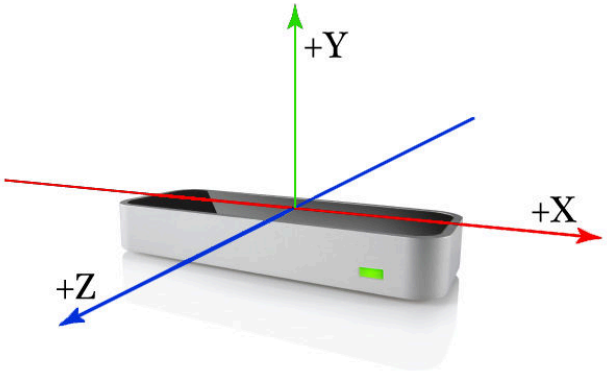


Figure 3. The Leap Motion right-handed coordinate system [30]

tion Inc and released in 2013 [26]. It recognizes movements of hands, fingers, and objects that resemble them and it is designed to perceive motion as users naturally move their hands [7]. Leap Motion has a sub-millimeter accuracy, allowing a position accuracy of 0.2 mm, tested with an industrial robot with a reference pen [27]. Although they are other motion controllers, e.g. the Kinect created my Microsoft (2013), the IC Air created by Steinberg (2013) or the iRing created by IK Multimedia, the reason behind selecting the Leap Motion was that comparable controllers in the same price range, were not able to achieve its accuracy [27].

The device operates at close range with high precision and high frame-rate and is also able to detect finger positions, palm orientation, gestures and movement. The controller connects via the computer's USB port and uses optical sensors and infrared light. The sensors are directed to the vertical axis with an opening of 150 degrees and the distance from the controller varies between 25 and 600 mm from the device. In terms of hardware, it consists of two cameras and three infrared leds. Infrared leds are able to detect wavelengths of 850 nm [28]. Thanks to its wide-angle lens, the controller has a large, inverted, pyramid-shaped interaction space (8 cubic feet).

The Leap Motion software combines its sensor data with an internal model of the human hand coping with challenging tracking conditions [30]. For the detection purposes, a right-handed Cartesian coordinate system is used, with the origin centered at the top of the Leap Motion Controller. The x- and z-axes lie in the horizontal plane, with the x-axis running parallel to the long edge of the device [30]. It uses an infrared stereo cameras as a tracking sensor. For data processing the controller applies advanced algorithms to the raw sensor data, which bypass the background and lighting, and perform analysis for the three-dimensional representation of the operator's hands.

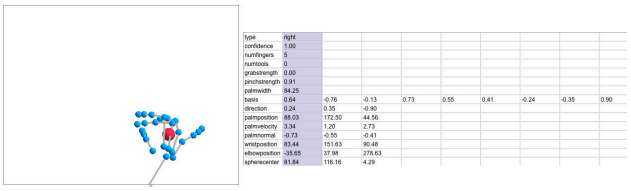


Figure 4. IRCAM's "Leap Motion Skeletal Tracking in Max" output data example

3.2 Skeletal Tracking

The "Leap Motion Skeletal Tracking in Max" patch was implemented by IRCAM [31] and was used for interfacing MAX/MSP with the motion controller. This patch takes as input the motion controller sensor data and outputs data collected from the motion controller (Figure 4). In particular, it returns 3D numerical data retrieved by the SDK of the leap motion controller, concerning the user's hands (position, direction, velocity, strength, etc), wrists and fingers (tip position, velocity, direction etc), as described in detail in the Leap Motion SDK [32, 33]. This information is displayed in the form of numeric variables in the MAX / MSP environment, usable by the user for the design and definition of novel gestures.

3.3 Gestures

New gestures had to be designed, since the ones recognised by the existing Leap Motion SDK Library (CircleGesture, KeyTapGesture, ScreenTapGesture, SwipeGesture, etc.) [30] were not compatible with the natural movement of the performer's body and hands. Most of them were designed to simulate the control of smartphones or tablets.

The gesture dictionary was designed and implemented so that a musician can easily recall and activate the desired musical intervals and operations. Each gesture functions as a "pose", or "a static gesture", as it activates or deactivates prespecified toggles, thus enabling easy control of Polyfōnía's live looping or harmonizer module which offers up to fourteen parallel voices. As per Mitra et al. [34] a gesture can be static or dynamic,Thus, for the sake of convenience we call them all simply "gestures".

By opening or closing both one's hands, users can activate or deactivate the system.

By closing one's left hand, users can activate the live looping module. The live-looping control gestures are designed to emulate a more natural movement of "grabbing" and "releasing" the sound.

Using one's right hand fingers, a user can enable the harmonizer module, thus creating musical intervals from minor second to major ninth. The position of the hand (to the left or right of the central axis of the controller, as described in Table 1), can indicate a smaller or a bigger musical interval. For the harmonizer processor, we set the performer's options to activate and deactivate voices via toggles (rather than keeping voices active throughout the duration of each movement). The motivation for a discrete-only control was the ability that multiple voices could be manipulated simultaneously with a few movements. In addition, the importance of the gestures' easy memorization and thus, the numbers of the musical intervals were used so that there can be a direct correspondence with the musical intervals.

Figure 5 shows the gesture vocabulary, Figure 6 shows the toggles that activate or deactivate each of 14 voices and Table 1 shows the suggested configuration, which is mentioned in the Evaluation section. However, it is important to mention that the performer can easily assign any gesture to any pitch shifting, simply by modifying the number of semitones for the chosen toggle in the Max/MSP patch, thus changing the musical interval that each gesture activates.

3.4 Polyfōnía Live Looping and Harmonizer Modules

The live looping processor consists of a live looper which permits live audio recording from the input microphone and play it on repeat, controlled by toggles. The harmonizer processor can perform pitch shifting on the input acoustic signal. It can pitch-shift the input signal to match the twelve halftones of the european twelve-tone scale, with a separate toggle switch to enable or disable each of the voices. The implementation of vocal processor components is not within the scope of the proposed work. For the Polyfōnía field testing we made modifications of basic Max/MSP audio processors originated from the Max/MSP's library and forum. However, the Polyfōnía's main interaction modules can work with any loop activated with toggles and any up-to-14-voices harmonizer, enabling and disabling each of its voices with a toggle.

3.5 Polyfōnía Interaction Modules

The Live-looping Interaction Module and the Harmonizer Interaction Module constitute the core of the system and the algorithm was implemented by the authors in Max/MSP. These two modules are responsible for the real-time interaction between the user and the parametrization of the digital processing units through the user's hand movements, as described in the previous section. They take the data from the Skeletal Tracking component as input, calculate the distances between all of the fingers' positions involved in each movement, and make the corresponding changes in the Vocal Processors. The variables used in this module include fingers' tip position, grab strength (the strength of a grab hand pose) and orientation of both user hands.

Semitones	Interval	Gesture (Side)
1	Minor second	II (LEFT)
2	Major second	II (RIGHT)
3	Minor Third	III (LEFT)
4	Major Third	III (RIGHT)
5	Perfect Fourth	IV (LEFT)
6	Augmented Fourth	IV (RIGHT)
7	Perfect Fifth	V (LEFT)
8	Augmented Fifth	V (RIGHT)
9	Major Sixth	VI (LEFT)
10	Augmented Sixth	VI (RIGHT)
11	Major Seventh	VII (LEFT)
12	Octave	VII (RIGHT)
13	Augmented Eight	VIII (LEFT)
14	Ninth	VIII (RIGHT)

Table 1. The correspondence between the gestures and the music intervals.

The Live-looping Interaction Module controls the live-looping processor (i.e. starts and stops playback of the recorded signal) with the predefined hand gestures. For the previous mentioned gestures recognition, we analysed the data regarding the intensity of the grab for both hands.

Regarding the handling of the Harmonizer Processor, the performer can enable or disable each voice switch with a predefined gesture (Figure 6). This module recognises seven movements, each of which activates a separate voice if formed to the left or right of the vertical axis starting from the center of the motion controller. Therefore, as shown in Table 1, the vocal performer can activate up to fourteen parallel voices, creating the desired harmony effect.

The implementation of Polyfōnía, and demo of its use, can be found here [1] [25].

4. EVALUATION AND PERFORMANCE

The system was tested on a MacAir 11-inch, macOS 10.13, with a 1,3 GHz Intel Core i5 processor, 4 GB RAM and an Intel HD Graphics 5000 1536 MB GPU. The connection was instant and without any technical difficulties, and the motion detector's response to the aforementioned operating system was satisfactory without substantial delays, despite the fact that system used was just above the minimum for the motion detector's recommendations.

The proposed configuration between gestures and pitch-shifted semitones (for the Harmonizer Processor) during the system's field testing, is shown in Table 1. The handling of the system, as proved in the demonstration, was natural and easy, reliable and with very few miliseconds of delay (20 ms), making Polyfōnía suitable for live performances. The demonstration video of the system can be found here [2] [25].

[1] drive.google.com/drive/folders/16n3seCv_5daH0azoDbSIosBYmIcZvdda

[2] drive.google.com/drive/folders/1eIiOdRANiiN_ax2FoaN4WS5_RXAI2Pe-

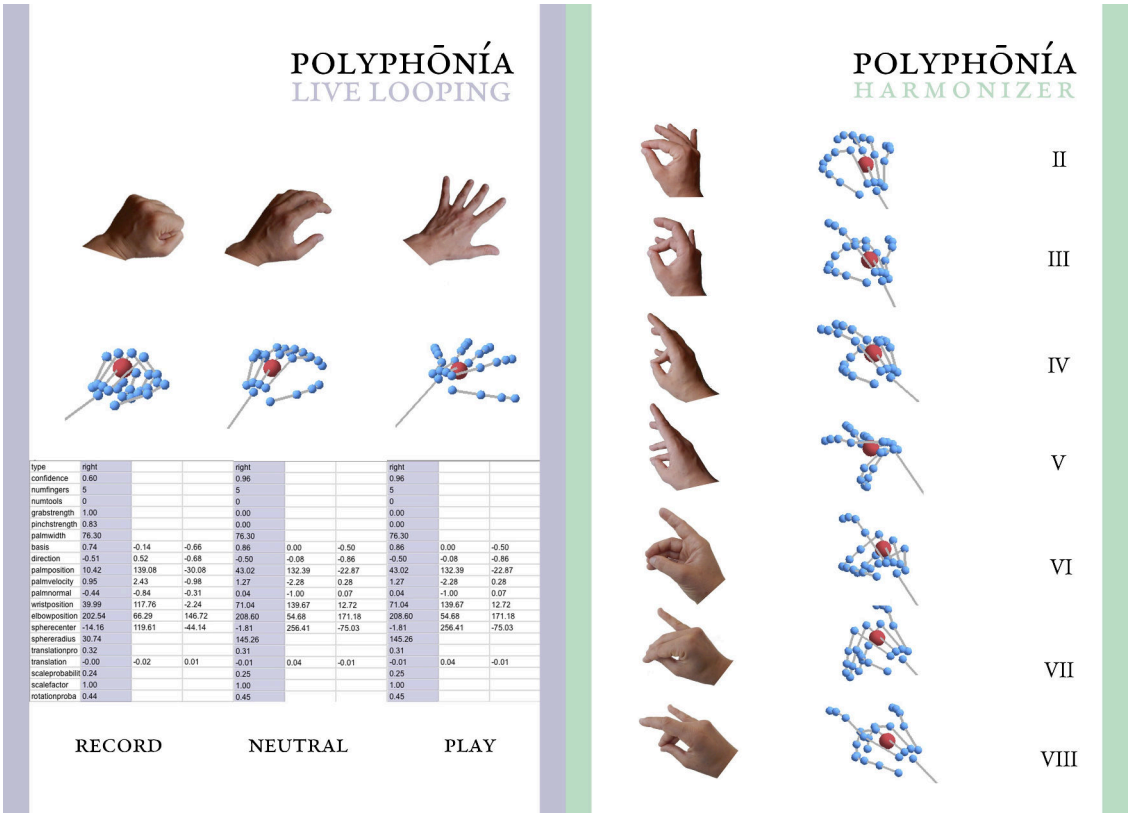


Figure 5. Polyfōnía modules' gestures

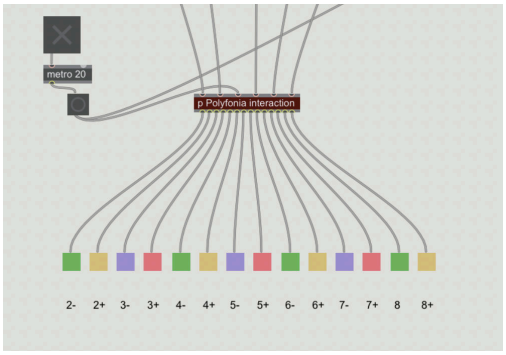


Figure 6. The Harmonizer Interaction module activates the fourteen voices toggles.

For the evaluation of the Polyfōnía's Live Looping Module, the gestures "Record" and "Play" (Figure 5) were evaluated, performing 100 repetitions for each gesture, achieving both 100% accuracy. For the evaluation of the Polyfōnía's Harmonizer Module, a confusion matrix was created. The system was intentionally tested in poor light conditions. Due to symmetry, only the left side was examined, for a right-handed user, for the seven gestures (Table 1, Figure 5) and the no-gesture (0). For each gesture 100 repetitions were performed. Figure 7 shows the confusion matrix. The overall accuracy of the Polyfōnía is 0.945, computed by dividing the total number of true positives over the total number of tests.

Polyfōnía was presented for a first time in a solo-perfomance, of the XXX Festival, funded by YYY. A video containing a part of this live performance can be found here [3] [25]. In the first live performance of the system, due to the lack of (in-ear) monitoring [35], the performer uses the screen for checking the loops and different voices activation. In similar cases, a projection of the feedback behind the public could be used. During the performance, both of the Polyfōnía modules were used (live-looping and harmonizer). The performance took place after the sundown, with stage lighting, and it was resonated by the audience in a very positive way. The response of the system was very good, as no problems or substantial delays were observed. The system can't follow extremely fast hand movements but it is fully suitable for use in vocal polyphonic singing, and for any performance that does not requires extremely quick changes in polyphonic singing patterns. As stated by a conductor which was present at the Polyfōnía's first performance, its delay may even seems natural, as it is similar to the natural response of a choir in a conductor's gestures.

5. FUTURE WORK

Polyfōnía could be enriched with new movements as well as with additional vocal processors. An interesting field of research is a kinesiological study for the control of many more parameters and functions, maintaining simplicity and

[3] drive.google.com/file/d/1YqPztgA1bfeLfBc5VZYrXo02fZjgwZCO/view

PREDICTED \ TRUE	II	III	IV	V	VI	VII	VIII	O
II	96	0	0	0	0	0	0	3
III	0	97	0	0	2	2	0	0
IV	0	0	97	0	4	0	0	0
V	0	0	0	89	0	0	0	0
VI	0	0	0	0	87	0	0	2
VII	0	0	0	0	0	91	0	1
VIII	0	0	0	0	0	0	96	2
O	4	3	3	11	7	7	4	92

Figure 7. Polyfōnía's confusion matrix

Figure 8. Performing with Polyfōnía.

clarity in its handling. Continuous control of the effects could be added, offering e.g. time shifting for the loops of the Live-looping Interaction module and volume or other effects automation for the Harmonizer Interaction Module.

The Live-looping Interaction module, could be developed to record more tracks enabling loop hierarchy manipulation. The Harmonizer Interaction Module could also be combined with an automatic harmonization system for specific music genres, e.g., a pretrained deep neural network. The perfomer could then control the degree of intervention of the automatic harmonization with gestures or set up the starting values. Moreover, a pretrained neural network in polyphonic music could also be embedded as an automatic harmonizing continuator, to take the performer's intended use of harmony as input in order to imitate the performer's aesthetics.

6. CONCLUSIONS

The control of vocal processors without pedals, knobs, gloves or wiring, may open up new horizons in vocal performance, enabling the use of processors in either demanding configurations, theatrical settings or expressive restrictions and requirements. The simplicity of Polyfōnía's use, coupled with the absence of equipment and combined with the sound enrichment that processors provide, can create a liberating contrast with a variety of applications and unexplored features.

7. REFERENCES

[1] E. Edmonds, "The art of interaction: What HCI can learn from interactive art," *Synthesis Lectures on Human-Centered Informatics*, vol. 11, pp. i–73, 03 2018.

[2] M. Jeon, R. Fiebrink, E. A. Edmonds, and D. Herath, "From rituals to magic: Interactive art and HCI of the past, present, and future," *International Journal of Human-Computer Studies*, 2019.

[3] G. Luck and P. Toiviainen, "Ideal singing posture: evidence from behavioural studies and computational motion analysis," pp. 15–19, 09 2007.

[4] F. F. Staes, L. Jansen, A. Vilette, Y. Coveliers, K. Daniels, and W. Decoster, "Physical therapy as a

means to optimize posture and voice parameters in student classical singers: A case report," *Journal of Voice*, vol. 25, no. 3, pp. e91 – e101, 2011.

[5] P. Hodges, R. Sapsford, and L. Pengel, "Postural and respiratory functions of the pelvic floor muscles," *Neurourology and urodynamics*, vol. 26, pp. 362–71, 05 2007.

[6] B. M Wilson Arboleda and A. Frederick, "Considerations for maintenance of postural alignment for voice production," *Journal of voice : official journal of the Voice Foundation*, vol. 22, pp. 90–9, 02 2008.

[7] M. Ritter and A. Aska, "Leap motion as expressive gestural interface," in *Proceedings ICMC, SMC, 2014*, 2014.

[8] R. Morales-Manzanares, E. Morales, and R. Dannenberg, "Sicib: An interactive music composition system using body movements," *Computer Music JournalArticle) Computer Music Journal*, vol. 25, pp. 25–36, 06 2001.

[9] D. Rodriguez and I. Rodriguez, "Real-time visual sound installation performed by glove-gesture," in *Proceedings of the 2005 Conference on New Interfaces for Musical Expression*, ser. NIME '05, Vancouver, Canada. Singapore, Singapore: National University of Singapore, 2005, pp. 252–253. [Online]. Available: http://dl.acm.org/citation.cfm?id=1085939.1086015

[10] C. Kiefer, N. Collins, and G. Fitzpatrick, "Phalanger: Controlling music software with hand movement using a computer vision and machine learning approach," 07 2019.

[11] C. Dobrian and F. Bevilacqua, "Gestural control of music: Using the vicon 8 motion capture system," in *Proceedings of the Conference on New Interfaces for Musical Expression*, ser. NIME. SGP: National University of Singapore, 2003, p. 161–163.

[12] G. E. Lewis, "The virtual discourses of pamela z," *Journal of the Society for American Music*, vol. 1, no. 1, pp. 57–77, 2007.

[13] E. N. Jessop, "The vocal augmentation and manipulation prosthesis (vamp): A conducting-based gestural controller for vocal performance." in *NIME*, 2009, pp. 256–259.

[14] J.-S. Gualtieri and P.-A. Bisgambiglia, "A polyphony learning software (pls) based on gesture recognition by fuzzy inference system," *Journal of Intelligent & Fuzzy Systems*, vol. 28, no. 4, pp. 1795–1803, 2015.

[15] L. Feugère, S. Le Beux, and C. d'Alessandro, "Chorus digitalis: polyphonic gestural singing," in *1st International Workshop on Performative Speech and Singing Synthesis (P3S 2011)*, 2011.

[16] R. Tavares, H. Mesquita, R. Penha, P. Abreu, and T. Restivo, "An instrumented glove for control audiovisual elements in performing arts." *International Journal of Online Engineering*, vol. 14, no. 2, 2018.

[17] S. Smallwood, P. Cook, D. Trueman, and T. McIntrye, "Don't forget the loudspeaker." in *NIME09, June 3-6, 2009, Pittsburgh, PA*, 2009. [Online]. Available: http://www.scott-smallwood.com/pdf/delorean.pdf

[18] F. Berthaut, M. Desainte-Catherine, and M. Hachet, "Drile: An immersive environment for hierarchical live-looping," in *NIME*, 2010.

[19] L. Hantrakul and K. Kaczmarek, "Implementations of the leap motion device in sound synthesis and interactive live performance," in *Proceedings of the 2014 International Workshop on Movement and Computing*, ser. MOCO '14, Paris, France. New York, NY, USA: ACM, 2014, pp. 142–145.

[20] K. Warren, "Composing and performing digital voice using microphone-centric gesture and control data," in *ICMC*, 2016.

[21] ——, "Sound technologies as agency-granting prosthesis to vocal body," *Leonardo Music Journal*, vol. 28, pp. 30–33, 2018.

[22] K. Norderval, "Electrifying opera: Amplifying agency for opera singers improvising with interactive audio technology," 2020.

[23] S. Eriksson, Å. Unander-Scharin, V. Trichon, C. Unander-Scharin, H. Kjellström, and K. Höök, "Dancing with drones: Crafting novel artistic expressions through intercorporeality," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–12.

[24] S. Eriksson, K. Höök, R. Shusterman, D. Svanes, C. Unander-Scharin, and Å. Unander-Scharin, "Ethics in movement: Shaping and being shaped in human-drone interaction," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–14.

[25] "Polyfõnía: Implementation, demonstration and live performance," https://drive.google.com/drive/folders/16n3seCv_5daH0azoDbSIosBYmIcZvdda?usp=sharing, 2021.

[26] M. Ritter and A. Aska, "Leap motion as expressive gestural interface," in *ICMC*, 2014.

[27] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, "Analysis of the accuracy and robustness of the leap motion controller," *Sensors*, vol. 13, no. 5, pp. 6380–6393, 2013.

[28] L. Motion. Leap motion sdk v2.3.1. [Online]. Available: https://developer.leapmotion.com/sdk/v2

[29] "Leap motion blog," https://blog.leapmotion.com/how-to-build-your-own-leap-motion-art-installation/leap-motion-interaction-area/.

[30] "Leap motion sdk," https://developer-archive.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html.

[31] IRCAM. (2014) Sound Music Movement leap motion skeletal tracking in max. [Online]. Available: http://ismm.ircam.fr/leapmotion/

[32] "Leap motion sdk: Hand," https://developer-archive.leapmotion.com/documentation/csharp/api/gen-csharp/class_leap_1_1_hand.htmll.

[33] "Leap motion sdk: Fingers," https://developer-archive.leapmotion.com/documentation/csharp/api/gen-csharp/class_leap_1_1_finger.html.

[34] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, 2007.

[35] N. Joll, *Philosophy and the Hitchhiker's Guide to the Galaxy*. Springer, 2016.

Zero-cost Abstractions To Manage Audio Resource Allocation In Functional Programming Languages

Mike Solomon
wags.fm
mike@mikesolomon.org

ABSTRACT

Audio programming languages fall into two broad categories: imperative (CSound, sclang) and functional (Faust, tidal). For the latter, the declarative and immutable nature of data and control structures are often at odds with the mutable and referentially opaque nature of audio units like oscillators and filters. Recently, languages such as Rust have bridged this chasm through zero-cost abstractions that manage resource allocation in otherwise functional settings. In the same spirit, this paper presents a zero-cost abstraction for the management of audio units when working with pure audio rendering functions. The result is automatic audio memory management that incurs a minimal runtime penalty while retaining an expressive functional syntax.

1. INTRODUCTION

The prevailing resource-management paradigm in all modern DAWs and most visual or text-based music creation environments is to model resources as units that are connected to each other. This is done via patcher chords in PD and Max, busses in SuperCollider, and argument passing in CSound.

Almost all audio units preserve some notion of state internally. For example:

- Waveforms remember their previous position in a lookup table so that changes in frequency do not disrupt the phase.
- Biquad filters and delay lines need to access a certain number of input samples in the past as well as potentially their own output.
- FFT-based algorithms need to access and aggregate previous FFTs using a windowing function before rendering the output in the time domain.

Because of this, substituting one audio unit for another cannot be done in the same way that one would substitute a referentially transparent value like a floating-point number or boolean.

Copyright: © 2022 Mike Solomon et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

In imperative audio programming languages, such as the Web Audio API or PureData, managing stateful units is a core feature of the language. Units are assigned to variables on which arbitrary side effects, like setting a frequency or a volume, can be performed:

```
const oscillator = ctx.createOscillator();
oscillator.type = "square";
oscillator
  .frequency
  .setValueAtTime(440, ctx.currentTime);
oscillator.connect(ctx.destination);
oscillator.start();
```

However, in declarative and graph-based languages, this is more challenging. Consider, for example, the following pseudocode. It is presented in a Haskell-like syntax, and in it and the following examples, `kp` is a boolean value that answers the question "is a key currently pressed?".

```
type KPT = { time :: Number, kp :: Boolean }
render :: KPT → AudioGraph
render { time, kp } =
  (
    if time < 5
    then sinOsc (time * 220)
    else if time < 10
    then sinOsc (time * 50)
    else highpass (playBuf "hi")
  ) `append`
  (
    when kp (sinOsc 880)
  )
```

We are now demanding much more of the audio engine:

- Even though we haven't specified it, in almost all circumstances we will want to preserve the sine-wave oscillator across the five-second mark to avoid phase disruption.
- We do not want the engine to spuriously tear down or create new oscillators, for example arbitrarily changing oscillator units before the 10-second mark.
- We want as little runtime accounting of audio units as possible, ideally precompiling an efficient kernel to minimize how much time is spent on our control thread.

As functional frameworks such as Tidal Cycles [1] and PureScript Wags [2] become more prevalent, it is important to maintain the helpful abstraction of graphs as a pure function of time while rigorously keeping track of active and inactive audio units with as little computational overhead as possible.

This paper proposes a zero-cost abstraction to manage resource allocation in functional audio programming languages combining the following three techniques:

1. Cofree comonads [3, 4], as popularized by Edward Kmett.
2. Existential quantification [5, 6], first introduced as a programming paradigm by Nigel Perry.
3. Indexed types [7], which can be used to control transitions between states at compile time.

The paper will give examples of all three concepts, showing how they can be combined into an audio-resource management strategy. It will end by showing the results of a benchmark conducted using PureScript Wags.

2. COFREE COMONADS

The first leg of our functional journey will explore cofree comonads, which can be used as a drop-in replacement for pure functions of time in effectful settings while providing the added advantage of retaining an internal state both at the type *and* term level. It is this state that will ultimately power our zero-cost abstraction.

To understand what cofree comonads are, it is helpful to first define the following concepts in order: the functor, the comonad, and finally the cofree comonad.

2.1 Functor

A functor is a computational context into which functions can be mapped. For example, if there is a function `add1` that adds 1 to an integer, it can be mapped to the context of an array of integers $[1, 2, 3, 4]$ to produce $[2, 3, 4, 5]$ or to the context of a homogeneous record $\{x : 1, y : 2\}$ to produce $\{x : 2, y : 3\}$. The standard definition of `map` is $\forall$ `a b. (a` $\rightarrow$ `b)` $\rightarrow$ `m a` $\rightarrow$ `m b`, mapping a function `a` $\rightarrow$ `b` to operate in context `m`.

Functors are ubiquitous in audio programming: for example, when we apply a function `transposeSemitone` to a sequence of notes, we understand intuitively that the transposition function is lifted into the context of the sequence even though a rhythmic sequence, pedantically speaking, cannot be "transposed" harmonically.

While functors provide a way to lift functions into contexts like arrays and records, they provide no way to extract values from those contexts, nor do they provide a way to recursively nest contexts. As we'll see in the next section, these are bread-and-butter operations of any realtime audio rendering engine, and this is where comonads come in.

2.2 Comonad

A comonad [1] w is a functor that comes equipped with two operations that fulfill two laws.

The operations are as follows:

- **extract**, which extracts a value `a` from the term `w a`.
- **duplicate**, which takes a term `w a` and "nests" or "projects" it into `w (w a)`.

These operations must satisfy the following laws, where `e` is `extract`, `d` is `duplicate`, and `wa` is an arbitrary comonad:

- `e (d (wa))` $==$ `wa`
- `map e (d (wa))` $==$ `wa`

The `map` in the second law is the `map` discussed previously for functors.

Even though most audio frameworks do not use comonadic terminology in their codebase, these two operations — extract and duplicate — are ubiquitous in audio programming. `extract` is the operation to get information like control-rate data or samples from a computational context. [2] `duplicate` is the operation that takes a context and computes its logical next iteration, which is almost always a rendering or event loop. [3] .

Event loops have an additional unique property that is not captured by comonads - for any one iteration, they are *functorial* in nature. That is, the loop is almost never simply executed: it often exists in a computational context that furnishes it with information like the current time and external information via transports like MIDI or OSC. So, if the resultant control-rate data or sample buffer is s and the loop's context is a functor f, then we have $w(f\ s)$, which is no longer a comonad of s but one of $f\ s$. To resolve this dilemma, we introduce the *cofree comonad*.

2.3 Cofree Comonad

`Cofree` is a data structure that is parameterized by two types: a functor `f` and a value `a`. The definition is:

```
data Cofree f a =
  Cofree a (f (Cofree f a))
```

In languages with strict evaluation, this recursive definition would cause a stack overflow, and in practice, the definition often uses a unitary thunk:

```
data Cofree f a =
  Cofree a (Unit → f (Cofree f a))
```

The `Cofree` constructor takes two arguments: a value `a` and a computational context `f` in which one can reason about the next $Cofree\ f\ a$. Note that the next `Cofree f a` need not actually exist [4] — event loops can enter into failure states for example, but most server-based programs (ie `scsynth`) are built around the concept of a potentially infinite event loop.

Cofree is a functor because it has the following instance for `map`:

[1] Comonads are not to be confused with their close cousins, monads. While monads are outside the scope of this article, they are the categorical dual of comonads and can be used to model a large class of computations.

[2] For example, in the SuperCollider code base, this is defined as a function called `Callback` in the file `SC_AU.h`

[3] For example, in the SuperCollider code base, this is defined in the file `SC_EventLoop.hpp`

[4] For example, a functor f can choose to ignore its argument a, as is the case with ie the $Proxy$ functor, `https://hackage.haskell.org/package/base-4.16.0.0/docs/Data-Proxy.html`.

```
map f (Cofree a b) =
  Cofree (f a) (map (map f) b)
```

It is a comonad because it has the following instances for `extract` and `duplicate`:

```
extract (Cofree a _) = a
duplicate c@(Cofree _ b) =
  Cofree c (map duplicate b)
```

Most importantly, `Cofree` can be "unwrapped" by returning the second argument:

```
unwrapCofree :: Cofree f a → f (Cofree f a)
unwrapCofree (Cofree _ b) = b
```

In an audio rendering engine, this gives us the next iteration of a rendering loop `f (Cofree f a)`. For example, if `f` is `(→) Time` (where `Time` is a positive floating-point number), then at each unwrapping of the loop, we "unlock" the next step by providing the current time. This is also called the generator pattern in imperative languages like Python and JavaScript, where `unwrapCofree` is analogous to `yield` and the `f` around `f (Cofree f a)` is discharged with the `next` function.

2.4 Revisiting Our Pseudocode

Let's revisit our pseudocode through the prism of cofree comonads. We'll reduce the size of the example for brevity:

```
type KPT = { time :: Number, kp :: Boolean }
render :: KPT → AudioGraph
render { time, kp } =
  (
    if time < 5
    then sinOsc (time * 220)
    else highpass (playBuf "hi")
  ) `append` (when kp (sinOsc 880))
```

We can rewrite this as a function that yields a cofree comonad. `extract` provides an audio graph, and `unwrapCofree` yields the next iteration of an event loop.

```
type KPT = { time :: Number, kp :: Boolean }
render :: KPT → Cofree ((→) KPT) AudioGraph
render { time, kp } =
  Cofree
    (
      (
        if time < 5
        then sinOsc (time * 220)
        else highpass (playBuf "hi")
      ) `append` (when kp (sinOsc 880))
    )
    render
```

Here, because the underlying Functor is `((←) KPT)`=, we can use `render` recursively in its own definition.

In practice, this also changes the way an event loop is written:

```
-- with pure functions
loop = do
  time ← getTime
  kp ← ketKp
  let res = render { time, kp }
  doSomethingWith res
  delay 0.03
  loop
```

```
-- with a cofree comonad
loop =
  let f r = do
    time ← getTime
    kp ← ketKp
    let res = render { time, kp }
    doSomethingWith (extract res)
    delay 0.03
    f (unwrapCofree res)
  in f render
```

So far, this formulation provides no advantage over a pure function of time. However, things get more interesting when we introduce arbitrary internal states in the definition of our cofree comonad. For example, if we want the sine-wave oscillator to release one second after a keypress ends, there is no way to do this with a pure function of time because there is no retention of state. Cofree comonads, on the other hand, make this trivial.

```
type KPT = { time :: Number, kp :: Boolean }
render :: KPT → Cofree ((→) KPT) AudioGraph
render  =
  let
    f kp_1 t_1 { time, kp } =
      Cofree
        (
          (
            if time < 5
            then sinOsc (time * 220)
            else highpass (playBuf "hi")
          ) `append`
          (when (kp || time < t_1 + 1)
                (sinOsc 880))
        )
        (
          render kp
            (if kp_1 && not kp
                then time else t_1)
        )
  in f false (-inf)
```

While this section has showed us how to write an event loop and how to extract `k`-rate/`a`-rate data from a computation in a functional style, it still has not solved the issue of resource management of stateful audio units. To do this, we will need to introduce the concept of existential quantification and use that to build a indexed type system within a cofree comonad.

3. EXISTENTIAL QUANTIFICATION

In the last example from Section 2, we saw how state can be curried in a cofree comonad's definition to create effects like delayed releases of oscillators and in general any sort of memory cell needed for filtering.

This technique of propagating a term internally through a computation is called a *closure*. The terms `t_1` and `kp_1` are *lexically closed* within the environment of the cofree comonad, but are never exposed to the external computation. This technique comes with one major drawback, however: all functions that operate on the internal state need to be defined within the closure, which makes it difficult to operate on the state using external APIs.

To remedy this problem, we can use existential types. We say the type is existentially quantified, or a coend [8], when

a consumer of the type knows only that it exists without knowing what the type is. As an example, the `Exists` type in PureScript can be used to turn any type constructor $f\ a$ into an existential type $\exists a.f\ a$.

```
data StreamF a s = StreamF s (s → Tuple s a)
type Stream a = Exists (StreamF a)
```

One can then get into and out of the existential type using the following two functions:

```
mkExists :: forall a f. f a → Exists f
runExists
  :: forall r f
  . (forall a. f a → r)
  → Exists f
  → r
```

A useful way to think about existential types is in terms of epistemic certainty. If a universally quantified definition is true "for all" types (this is the norm in functional programming languages, for example the $identity$ function is defined as $\forall a.a \to a$), then it is a tautology. On the other hand, existential types are the weakest epistemological claim one can make: they assert that something exists without knowing anything about it.

We can use existential types to transform the closure from our previous render loop into an API that exposes an internal state as type `c` to external consumers. This new type is called `WithCtrl`, and existential types are used to transform it into a cofree comonad that the rendering loop understands.

```
type WithCtrl f a c = Cofree f (a /\ c)
type Ex f a = Exists (WithCtrl f a)
type KPT = { time :: Number, kp :: Boolean }
render
  :: KPT
  → WithCtrl ((→) KPT) AudioGraph KPT
render  =
  let
    f n t = Cofree
      (
        ((
          if n.time < 5
          then sinOsc (n.time * 220)
          else highpass (playBuf "hi")
        ) `append`
        (when (n.kp || n.time < t.time + 1)
              (sinOsc 880))) /\ t
      )
      (
        render n.kp
          (if t.kp && not n.kp
              then n.time else t.time)
      )
  in f false (-inf)

asEx :: KPT → E ((→) KPT) AudioGraph
asEx = mkExists <<< render

unEx
  :: E ((→) KPT) AudioGraph
  → Cofree ((→) KPT) AudioGraph
unEx = runExists (map fst)
```

Closures and existential types are particularly useful in recursive definitions, like those of cofree comonads, because of their ability to change their internal type over the lifetime of a looping control structure. For example, a state can be a `Number` followed by a `Boolean` followed by a `String` over the course of a computation instead of a union `Number | Boolean | String`. This significantly improves runtime performance (no pattern matching) and makes it easier to reason about programs, as nonsensical data (ie a filter's `q` value being passed to a sine wave oscillator) is prohibited at the type level. It is also a necessary stepping stone to indexed types, which is the crux of our resource-management strategy in Section 3.

As a motivating example, let's use a render function that switches from an oscillator and a filtered sound at the five-second mark. The behavior of the oscillator and of the filtered sound change irreversibly once a key is pressed. As a first pass, we'll use closures to implement this behavior.

```
renderHP kp_1 fq { kp } =
  let
    isKp = kp_1 || kp
    f = fq.f + (isKp ? 0 : 1000)
    q = fq.q + (isKp ? 0 : 40)
  in Cofree
    (highpass {f: f, q: q } (playBuf "hi"))
    ( render isKp fq)

renderS kp_1 f { time, kp } =
  if time > 5
  then renderHP kp_1 {f:900,q:1} {time,kp}
  else
    let
        isKp = kp_1 || kp
        f = f.f + (isKp ? 0 : 200)
      in Cofree
        (sinOsc {f: f})
        (renderS isKp fq)

render = renderS false {f: 440}
```

The types of the two internal states, `fq` and `f`, are different: one (used in `renderS`) contains frequency information for an oscillator whereas the other (used in `renderHP`) contains frequency and q values for a filter. And yet, they can be used inside the closure with no runtime penalty (no pattern matching) and no danger of accidentally passing information for the filter to the oscillator or vice versa.

It is possible to pass from closures to existential types by using the continuation passing pattern [9], whereby the control flow of a function (in our case, the loop) is yielded to a continuation. In the following example, for the sake of brevity, we will use monomorphic versions of the full type called `F` and of the existential type called `S`. The `AudioGraph` type is renamed `A`, and our continuation passing function is called `cp`. [5]

```
type C = {t :: Number, kp :: Boolean }
data E a b = L a | R b
data M a = J a | N
data U = U
-- F a is our type that will be existentially
-- quantified
data F a = F A a
-- S is our existential type that will erase
-- an a from (F a), leaving just the A term
data S = S (C → Tuple A S)
```

[5] Note that, in this example, lazy evaluation is assumed. In strict languages, it causes a stack overflow and the construction of the cofree comonad would need to be deferred.

```
-- continuation passing instead of a direct
-- cofree comonad constructor
cp
  :: (C → E S (F a))
  → (F a → S)
  → Scene S
cp m t = S go
  go time = case m time of
    L s → (\(S x) → x) s env
    S f@(F c a) →
      Tuple a (t f)

branch
  :: (F a → C → E S (F a))
  → F a
  → S
branch fa w = cp (fa w) (branch fa)

freeze :: F a → S
freeze s = cp (\_ → R s) freeze

-- our internal type will be the onset of
-- the keypress
frame0 { t, kp } =
  F (sinOsc 440.0) (kp ? J t : N))

loop (J t0) { t, kp } =
  F (sinOsc 440.0 + (t - t0)) (J t0)
loop N c = frame0 c

rest = highpass (playBuf "hi") /\ U

render =
  cp
    frame0
    (branch \(F _ a) c
        if t < 5
        then R (loop a c)
        else L (freeze rest)
```

Now, it is possible to define functions that interact directly with the `F a` contained by `frame0`, `loop` and `rest`, all of which contain an internal state — `M C` in the case of `frame0` and `loop`, `U` in the case of `rest`. For example, to override the internal state of `frame0` to always be `N`, we could write `frameN = (map (`
`(F c _) → F c N) frame0)`.

Furthermore, we have abstracted time management to the top-level `render` function, separating temporal branching from the definition of audio behavior. The function `cp` takes care of the type erasure from `F a` to `S` in the same way that `Exists` erases a type.

As we saw with the `frameN` example above, existential types can be used both to model the type-level morphology of internal states and to write APIs for interacting with these states. It also the crucial step that allows us to reason about resource management, as we will see in the next section on indexed types.

4. INDEXED TYPES

Let's take a closer look at the definitions of `F a` and `S` from the previous example. To create `S`, we drop both the internal type `a` *and* the term of type `a`. Instead, we propagate both the term and the type through the computation, allowing it to change as the computation progresses.

Nothing stops us from erasing multiple types, nor do all of the types have to appear in the corresponding terms. For example, we could have defined `F` as `data F i a = F A a`. Here, `i` is a phantom type: in addition to being erased in `S`, it has no representation at the term level `F`. We can use types like these to track aspects of our program at compile time. This is called a "zero-cost abstraction": it is an abstraction that offloads computationally expensive algorithms like graph traversals to the compiler. At runtime, there is no performance penalty.

When phantom types are used in this way, we call them *indexed types*. Indexed types are the key to managing the allocation of resources of an audio graph using cofree comonads. Changes in indices represent changes in the underlying audio graph. For example, if index i_0 is `SinOsc` and index i_1 is `Highpass (PlayBuf)`, the compiler knows that it will have to emit instructions to destroy a sine wave oscillator and create a playable buffer linked to a highpass filter.

Let's see this technique applied to the same example above where a sine-wave oscillator changes to a filtered buffer.

```
type C = {t :: Number, kp :: Boolean }
data E a b = L a | R b
data M a = J a | N
data U = U
data F i a = F A a
data S = S (C → Tuple A S)

eg2s :: F EmptyGraph a → F SinOsc a
eg2s = unsafeCoerce

s2s :: F SinOsc a → F SinOsc a
s2s = unsafeCoerce

s2h :: F SinOsc a → F (HighPass (PlayBuf)) a
s2h = unsafeCoerce

h2h
  :: F SinOsc a
  → F (HighPass (PlayBuf)) a
h2h = unsafeCoerce

cp
  :: (C → E S (F a))
  → (F a → S)
  → Scene S
cp m t = S go
  go time = case m time of
    L s → (\(S x) → x) s env
    S f@(F c a) →
      Tuple a (t f)

branch
  :: (F a → C → E S (F a))
  → F a
  → S
branch fa w = cp (fa w) (branch fa)

freeze :: F a → S
freeze s = cp (\_ → R s) freeze

frame0 { t, kp } =
  e2s $ F (sinOsc 440.0) (kp ? J t : N))

loop (J t0) { t, kp } =
  s2s $ F (sinOsc 440.0 + (t - t0)) (J t0)
```

```
loop N c = s2s $ frame0 c

rest = highpass (playBuf "hi") /\ U

render =
  cp
    frame0
    (branch \(F _ a) c
        if t < 5
        then R (loop a c)
        else
          L
          $ cp
            (\_ → s2h rest)
            (\_ → freeze (h2h rest))
```

This example is more complex in that there are four transitions between indexed states:

1. At time 0, a transition from the empty graph to a sine-wave oscillator (`e2s`).
2. At all remaining times until five seconds, a sine-wave oscillator to itself, meaning a no-op (`s2s`).
3. At the five-second mark, transition from a sine-wave oscillator to a filtered buffer (`s2h`).
4. From here until the audio is shut off (`h2h`).

On the term level, these transitions are no-ops that are tracked entirely at compile-time, which speeds up audio rendering. Instead, the only changes that need to be propagated to the graph are new control values, which requires no diffing between graphs. Furthermore, the use of control structures like `sinOsc` and `playBuf` can be verified against their type-level counterparts so that control data for an oscillator is never allowable when an oscillator is not in the type-level graph, etc.

For several examples of this technique being used in actual audio computations, one may consult the example folder of the `purescript-wags` project [6].

Before looking at benchmarks in the next section, let's revisit the path to indexed types to show how the allocation of audio resources can be tracked by indices in a purely functional setting.

1. Instead of repeatedly calling a pure function of time, rendering loops can unfold a cofree comonad that yields functions of time at each iteration. This is also called a Moore machine.
2. Cofree comonads can have changing internal states, and these states can be represented at the type-level as multivariate type constructors that are transformed into cofree comonads via existential quantification.
3. One of these existentially-eliminated types can be an index that tracks the current state of the audio graph. The compiler, as it tracks the evolution of this type, can emit efficient instructions for graph manipulations at compile-time, eliminating the need for runtime graph traversals.

[6] `https://github.com/mikesol/purescript-wags/tree/main/examples/kitchen-sink`

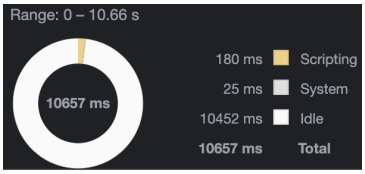


Figure 1. The rendering result of an audio graph created using zero-cost abstractions at compile-time.

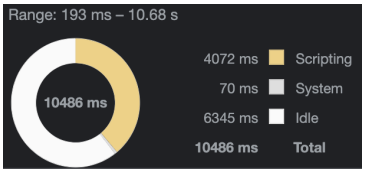


Figure 2. The rendering result of an audio graph created using runtime graph traversals.

5. EXAMPLE AND BENCHMARK

To illustrate the runtime benefits of this approach, the following test was run.

1. An audio graph with 61 units - 1 speaker, 30 gain controls and 30 sine-wave oscillators, was created twice using the Google Chrome implementation of the Web Audio API. Both versions model an audio graph as a pure function of time. [7]
   1.1. The first version [8] uses the zero-cost abstraction described in this paper.
   1.2. The second version [9] uses a runtime graph traversal to detect changes in the graph.
2. Approximately ten seconds of audio was rendered using the both versions.
3. The results are as follows:
   3.1. The first version's audio rendering took approximately 1.7% of the total rendering time. This is shown in Figure 1.
   3.2. The first version's audio rendering took 38.8% of the total rendering time. This is shown in Figure 2.

This shows that, for a moderately complex audio graph, the zero-cost abstraction was over 22 times faster than the version that made runtime graph traversals. This result can be reproduced at the following url: `https://smc2022.surge.sh`.

[7] The function can be seen at the following URL: `https://github.com/mikesol/purescript-wags/blob/b6930c5aab6ec5f04f7d8d07231a04b12f70388e/examples/smc2022/SMC2022.purs#L39`.

[8] See this url: `https://github.com/mikesol/purescript-wags/blob/b6930c5aab6ec5f04f7d8d07231a04b12f70388e/examples/smc2022/SMC2022.purs#L74`.

[9] See this url: `https://github.com/mikesol/purescript-wags/blob/b6930c5aab6ec5f04f7d8d07231a04b12f70388e/examples/smc2022/SMC2022.purs#L77`.

6. CONCLUSION

As functional audio programming becomes more ubiquitous and as large media houses like Epic Games [10] turn to functional programming to power parts of their rendering pipelines, the need for efficient representations of purely functional audio graphs becomes increasingly important. This paper presents a zero-cost abstraction for managing audio units by using existential typing to erase indexed representations of audio graphs in a cofree comonad. The resulting algorithm was able to outperform a runtime graph traversal by a factor of 22 on a moderately complex graph. While the algorithm was presented in Haskell-family pseudocode, the technique can be applied to any language that can perform type-level functions on partially-applied types, as `Exists` does in this paper (Scala, F#, PureScript, Agda, and Idris all have this feature, for example). By following this method, artists can benefit from the conceptual elegance and power of the functional style while enjoying the fast runtime behavior traditionally associated with imperative languages.

7. REFERENCES

[1] "Tidal cycles." [Online]. Available: https://tidalcycles.org/

[2] "Wags documentation." [Online]. Available: https://docs.wags.fm/

[3] "free: Monads for free," https://hackage.haskell.org/package/free-5.1.7, accessed: 2022-01-12.

[4] P. Freeman, "Declarative uis are the future—and the future is comonadic!" 2017.

[5] N. Perry, "The implementation of practical functional programming languages," Ph.D. dissertation, Citeseer, 1991.

[6] K. Läufer and M. Odersky, "Polymorphic type inference and abstract data types," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 16, no. 5, pp. 1411–1430, 1994.

[7] C. Zenger, "Indexed types," *Theoretical computer science*, vol. 187, no. 1-2, pp. 147–165, 1997.

[8] N. Yoneda, "On ext and exact sequences," *J. Fac. Sci. Univ. Tokyo Sect. I*, vol. 8, no. 507-576, p. 1960, 1960.

[9] G. J. Sussman and G. L. Steele Jr, "Ai memo no. 349 december 1975," *contract*, vol. 14, no. 75-C, p. 0643.

[10] See `https://discourse.haskell.org/t/an-epic-future-for-spj/3573`

Making Computer Music on the Web with JSPatcher

Shihong Ren
Shanghai Conservatory of Music, SKLMA, China
Université Jean Monnet, ECLLA, France
shihong.ren@univ-st-etienne.fr

Laurent Pottier
Université Jean Monnet, ECLLA, France
laurent.pottier@univ-st-etienne.fr

Michel Buffa
Université Côte d'Azur, CNRS, INRIA, France
michel.buffa@univ-cotedazur.fr

Yang Yu
Shanghai Conservatory of Music, SKLMA, China
yuyang@shcmusic.edu.cn

ABSTRACT

Web technology through the internet and a web browser, is attractive to modern computer music artists as it provides high accessibility and interactivity and opens to possibilities. Since the development of JSPatcher [1], an online visual programming language (VPL) for audio processing and interactive web-based programs, we have recently added various music-related features to the application, including important features that common music computing practices need, such as file management, audio buffer handling, audio plugin support, and computer-assisted composition functions. It aims to make it easier for musicians, composers or designers of multimedia projects who might be familiar with VPLs on native platforms like Max [2], PureData [3] or OpenMusic [4], to bring or create their work on the web with less effort. This paper presents the concept, implementation details and examples of these newly added features.

1. INTRODUCTION

JSPatcher is a web-based visual programming language (VPL) originally designed for providing a user interface (UI) for Web Audio API. [1] Since the API describes the audio processing flow as a graph of DSP nodes, it is convenient to have a *patcher* editing system [5] to manipulate the audio graph. JSPatcher is initially a WebAudio patcher editor that runs in a browser, where users can create boxes representing the DSP nodes and cables representing the connections in a canvas.

Since the whole JSPatcher platform is mainly developed in TypeScript and compiled to JavaScript which is the scripting language for the Web API, we have the possibility to fully import the language it self to the patcher system. Thus, in addition to the audio connection layer, in order to control DSP parameters with non-audio data, we added a dataflow layer that can distribute events between functions in real time. In this layer, various functions imported from the web environment, including JavaScript language built-ins and Web APIs, are available. These can be used to access the browser's high-level APIs, such as those for managing mouse and keyboard events, battery life or computer peripherals.

With these two layers in a single *imperative patcher*, the usage becomes similar to some VPLs available on native platforms like Max or PureData, while offering more flexible computational possibilities, taking advantages from the web community, as most of the JavaScript packages can be imported and used as functional boxes in a patcher.

An additional patcher interpretation mode, FAUST *compiled patcher*, has also been developed to facilitate DSP design under the WebAudio AudioWorklet specification [6]. The FAUST [7] language ecosystem is used to transform patchers under a specific mode into FAUST code which can be compiled to a customized DSP. This kind of can be used in imperative patchers as a special WebAudio node called AudioWorkletNode that can be connected with other WebAudio nodes. With this interpreter, some Max's Gen patchers can also be interpreted to FAUST code and be compiled in the same way in JSPatcher.

These new features make JSPatcher not only a utility to interface graphs from the Web Audio API, but also an integrated development environment (IDE) for data computation and audio processing on the web. Various possibilities related to music composition and performance are thus open for exploration on the platform.

The following sections will present the improvements made to JSPatcher that contain important features for modern computer music practices, including a file manager, audio buffer manipulation, audio plugin support and computer-assisted composition functions.

2. FILE MANAGER

2.1 Concept

One major limitation of the web environment is the access to the device's local files due to security reasons. According to the current web standards, web applications do not have permissions by default to read local files unless users explicitly allow them. The lack of a proper file manager API makes any web application difficult to maintain the structure of a file-based project.

To support sub-patchers and audio file manipulation in

[1] https://www.w3.org/TR/webaudio/

Copyright: © 2022 Shihong Ren et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Figure 1. JSPatcher with a file manager

JSPatcher, we had to design a virtual file system in the browser that allows the following operations:

1. Make the file system persistent after closing the application,
2. Upload files from the user's machine,
3. Create new files,
4. Delete files,
5. Copy or move files,
6. Access file data using its path.

Designing and implementing a persistent file storage system is a challenge in this work as there are few ways to keep reusable data in the browser. Our solution relies on IndexedDB, [2] a technology for client-side storage of significant amounts of structured data. Normally, in 2022 and on a desktop browser, up to two gigabytes of storage quota are available in the IndexedDB per site group, [3] . It is sufficient for any lightweight project. We also use BrowserFS, a JavaScript module that can store persistently a whole structured file system into the IndexedDB and provide a JavaScript API for file management [8, 9].

To accelerate file reading and writing in runtime, we also built a higher-level file system that caches the file tree structure and data in memory. It acts as a bridge between the BrowserFS and the UI, when a change is made by the user to the file system, it calls the BrowserFS to store new change in IndexedDB; meanwhile, all event subscribers to the changed file, such as the file manager UI (Figure 1), file editors or audio players will be alerted and they can react in real time.

2.2 Temporary File System

Along with the persistent file system in IndexedDB, we also implemented another memory-only file system for temporary entries. It is internally a label and file content map with event emitters and an observation mechanism.

The purpose of the temporary file system is to allow users to create memory spaces and use them with an associated name just like a persistent project file. A similar behavior exists in Max. For example, when a user defines an audio buffer using a name that doesn't refer to a file from the hard disk, the system will allocate a part of the memory that is temporarily accessible with this name. Any object that uses the same name will then refer to the same part of the memory. When every object associated with this name is removed, the memory will be freed and erased.

In JSPatcher, this behavior is implemented by the temporary file system in which every file is under the root directory. We have provided an API for objects to create temporary files when a name cannot be resolved as a persistent file. Meanwhile, the object becomes an observer of the temporary file. When the last observer gets removed, the temporary file will automatically free the memory space and remove itself.

2.3 File Structure of a Project

In JSPatcher, a project is a folder that contains subfolders or files in any format as in native operating systems. When a session is started, the project folder will automatically be initialized with a hidden file named `.jspatproj` under the root folder. This file contains metadata related to the project such as its name, its author, its dependencies and will be saved with the whole project.

Typically, a project will have one or multiple patchers with some asset files. To distinguish patchers under different modes, we use `.jspat` as the extension for regular imperative patcher, and `.dsppat` for FAUST compiled patcher. Patcher files can be uploaded to the file system or downloaded (exported) from it. Users can also upload or download the whole project for further exchange under the `.zip` format with every file compressed inside.

2.4 Usage in Patchers

Some types of project files can be recognized by JSPatcher and can be loaded into patchers. Typically, they will be used in a JavaScript patcher (using imperative and WebAudio layers) by different box objects. To load a specific file, the user should put the file path in Unix format (using slash for sub-directory) relative to the project root folder.

Table 1 shows a list of file types that can be loaded by box objects in a patcher. Figure 2 shows a sub-patcher file, an audio file, a text file and an image file in a patcher.

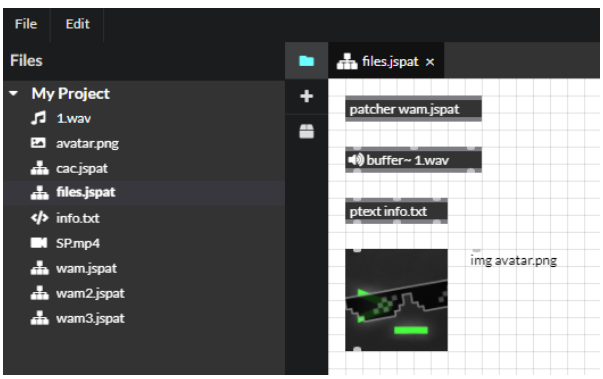


Figure 2. Files in a patcher

[2] `https://www.w3.org/TR/IndexedDB/`
[3] `https://tinyurl.com/muc8cmyx`

Extension	Loader	Description
`jspat`	`patcher`	Sub-patcher
`dsppat`	`pfaust`	Faust sub-patcher that can be compiled to an AudioWorklet DSP
`gendsp`	`gen`	Gen sub-patcher that can be compiled to an AudioWorklet DSP
Audio files (`wav`, `aif`, `mp3`, etc.)	`buffer~`	Audio files that can be decoded to in-memory audio buffers for further manipulations
Image files (`jpg`, `png`, `gif`, etc.)	`img`	Display the image in the patcher
Text files (`txt`, `json`, etc.)	`ptext`	Edit the text in the patcher

Table 1. A list of file types that can be loaded by box objects in a patcher.

3. AUDIO EDITING

3.1 Concept

Different kinds of web-based audio editors or digital audio workstations (DAW) have been developed by the community in recent years to bring desktop audio production experience on the web platform. However, for a long period, due to the inconsistent implementation of the Web Audio API between different web browsers, it has been hard to maintain related projects and ensure the same user experience across browsers and devices.

Support for the AudioWorklet API on Safari Desktop and iOS browsers was added in April 2021, which as the final step in providing full Web Audio API coverage on all major browsers. Also, the Web Audio API version 1.0 became, after ten years of maturation, a W3C recommendation (a frozen standard) in June 2021. These two events mark a milestone of the standard and its implementation process. It motivated developers to adapt Web Audio applications to the recommended standard without worrying about compatibility issues. The AudioWorklet API is important for audio editors or DAWs as buffer recording, looping, editing regions require sample-accurate controls over the input signal and can be done in a customized DSP through the API.

In JSPatcher, using the recent API, two features related to audio files were added. Audio files can now be edited in non-real time from a single-track audio editor and can be manipulated as an audio buffer in real time in a patcher.

3.2 Audio Editor

In the workflow for some genres of electronic music, especially *musique concrète* or interactive electroacoustic music, raw audio materials need to be cut and "cleaned up" before being put into a DAW project or a program. This process requires a tool that can directly modify the audio file with the following features:

1. Visualize any part of the waveform with rulers,
2. Play any part of the audio clip,
3. Select on the waveform with sample-accuracy,
4. Record audio in-place or after the clip,
5. Cut, copy and paste any part of the clip,
6. Adjust the volume of the selected section,
7. Silent the selected section or insert silence of any length,
8. Perform a phase inversion or reverse the selected section,
9. Fade-in and fade-out of any length with a flexible curve,
10. Resample the audio to any sample rate,
11. Create, delete, reorder channels, up-mix and down-mix (i.e., mix a stereo audio to a mono one),
12. Apply audio effects,
13. Export the audio in some formats.

Its implementation in JSPatcher is a dedicated window that can be opened by double-clicking an audio file listed in the file manager. (Figure 1)

When the user needs to open the editor, the system will firstly decode the given file to raw PCM data. Then, the data will be grouped into different levels of zoom that contain minimum and maximum values of each 16, 256, 4096 samples, etc. to optimize the waveform display for long audio files.

The layout of the editor's UI is similar to some desktop audio editors. A navigation bar with the waveform of the whole file is shown at the top of the window with a range of currently displayed and selected sections. Below the bar, a larger waveform viewer shows a section of the audio. Conventional features were implemented such as the cursor, the auto-scalable timing grid (vertical), energy (in dB) grid (horizontal), buttons to enable or disable channels for replay, fade-in and fade-out handlers, and an additional popup handler to adjust the gain when a section is selected.

Below the main viewer, a playback toolbar is presented. Users can play, pause or stop the playback from the cursor's position. Loop can also be enabled or disabled here. To record audio, users can choose an input device from a menu, enable the monitoring, and start recording using the red button.

At the bottom, a meter shows the current volume of the playing or the monitoring audio. A table with adjustable numbers shows the current displayed and selected section, and allows users to change it manually.

In the menu, a set of options are available to perform simple processing like silence, reverse, phase inversion, insert silence, resample, remix channels and export. To perform more complex processing using third-party DSPs, it is possible to load WebAudioModule (WAM) plugins [10, 11] into a plugin rack. Users can choose to process the whole audio file or only the selected part. We will present the WAM support in the next sections of the paper.

In the "Remix channels" option, users have access in a popup window to an I/O matrix to decide the number of outputs and how much volume of signals coming from

original channels for the resulting audio. (Figure 3)

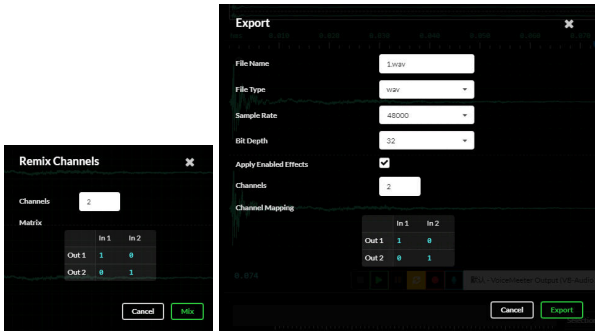


Figure 3. "Remix channels" and "Bounce" options

The audio export ("Bounce" option) supports `.wav`, `.mp3` and `.aac` formats with different sample rates or bit-rates thanks to a web implementation of ffmpeg [4] through Emscripten [5] compilation and WebAssembly. [6] [7] (Figure 3)

3.3 Audio Buffer Manipulation in a Patcher

Audio buffer manipulation is a key feature for any audio programming environment like Max or JSPatcher, as real time audio projects like some interactive music pieces often need to record and replay audio clips during the performance.

In the previous section, we mentioned that the `buffer~` object can refer to any audio file in the project or can create any temporary audio clip in the memory. The audio editor is also available when the user double-clicks the object. In addition, the editor can be "docked" beside the patcher.

The `buffer~` object accepts 4 possible arguments. The first argument is the identifier of the audio file or a temporary audio clip. If it is temporary, users can initialize it by specifying the buffer's number of channels, length in samples and its sample rate.

In fact, the object is associated with a PatcherAudio API in which the audio buffer is stored with its waveform data, with a set of methods related to the buffer editing. A Bang (a triggering event) from the first inlet of the `buffer~` object will trigger output of the PatcherAudio API.

Users can connect the `buffer~` object to some other objects to use the PatcherAudio API. For example, the `waveform~` object can display the waveform of the audio clip, `bufferSource~` object is a player that wraps the BufferSourceNode from the Web Audio API that accepts the PatcherAudio as the input and can play the audio with loop and different playback speed.

The PatcherAudio API provides necessary information getters like its number of channels, length and sample rate; and manipulation methods like split, concatenate, pick, paste, remove, insert, etc.

Figure 4 shows an example of picking a section (from sample 24000 to 25000) of the existing buffer with 48000 samples.

[4] `https://www.ffmpeg.org/`
[5] `https://emscripten.org/`
[6] `https://webassembly.org/`
[7] `https://github.com/ffmpegwasm/ffmpeg.wasm`

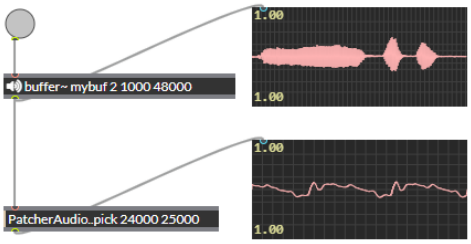


Figure 4. Audio buffer manipulation

4. AUDIO PLUGIN SUPPORT

4.1 WebAudioModule Plugin Standard

In the music production industry, most hardware devices have been substituted by software solutions over the past decades. Using DAWs with third-party audio plugins that act as synthesizers or audio effects is a common workflow for modern audio projects. VST (Virtual Studio Technology), introduced by Steinberg in 1996, is one of the most used cross-platform native plug-in formats based on C++. It provides a public SDK and API documents to allow plugin providers and host developers to implement them correctly.

Due to the limitation of the web platform, native audio plugins such as VSTs cannot be used in a web application without installing additional native software. Thus, for web-based DAWs, the need arose for a new standard of WebAudio plugins, offering similar functionality to their native counterparts.

In 2015, the first version of the Web Audio Modules (WAM) [10] standard is created, primarily for native plugins developers to port their existing plugins to the web. In 2018, they joined forces with other groups of people working on interoperable Web Audio plugins and plugin hosts to synchronize their efforts toward the beginnings of an open standard called Web Audio Plugins (WAP) [11, 12], covering a wider range of use cases. Recently, taking into account previous developments and the feedback received from developers over the past few years, a new version of the WAM standard has been created by emerging recent technological developments.

Now, a WAM plugin can be fetched from the web using a URI, and be initialized using the current WebAudio context. The plugin comes with a standardized API that can create a UI, get or adjust parameters, schedule events like parameter automation or MIDI messages, and connect with other WAMs or native WebAudio nodes.

4.2 WAMs in the Audio Editor

The Audio Editor in JSPatcher has a dedicated effect "rack" for WAMs. When the audio is loaded, the "rack" is empty so that users can add WAMs from their URIs. The audio will be played and processed through a pre-gain, then through the ordered WAM effects, finally through a post-gain. Users can use the rack UI to add or delete WAMs or display the WAM's own UI. (Figure 5)

Web Audio's OfflineAudioContext is an audio context

that, instead of process audio in real time, generates audio data from a WebAudio graph as fast as it can. When a user needs to export the entire audio file rendered with effects, or to apply the effects in-place, we will use a copy of the current "rack" in a new OfflineAudioContext to render the audio.

Figure 5. WAM plugins in the audio editor

4.3 WAMs in Patchers

It is also important to be able to create interactive audio programs with WAMs in JSPatcher. Using the `plugin~` object, users can load WAMs via a URI into a patcher.

When the `plugin~` receives a text string (as URI), it will remotely download the code from the URI and initialize it as a WAM. Then, it will automatically wrap the WAM and load its UI. Its inlets and outlets will be connectable with other WebAudio node objects. Additional inlets will be created to accept real time parameter change messages.

As WAMs support MIDI messages as input, the first inlet of the object accepts numbered arrays as MIDI event messages to the WAM inside. In addition, as WAMs have the ability to transmit non-audio events between each other, the patcher connection between two WAMs will be treated as a special one and make the necessary connection.

Figure 6 is an example of a WAM loaded in a patcher with an audio player.

Figure 6. WAM with an audio player in a patcher

5. COMPUTER-AIDED COMPOSITION

5.1 Context

Algorithmic composition often needs computer programs to calculate its musical representation. To design such a computer-aided composition (CAC) system for composers, visual programming and musical score display are two key features. OpenMusic, developed at IRCAM, shows the power of such a VPL in algorithmic composition with its composer-oriented design. The web environment, with its strong accessibility from device to device and flexibility from the UI perspective, is already a common platform for the low-code development system. It is likely to be highly suitable for a CAC system.

High-quality digital musical scores can be dynamically rendered and displayed on a computer using the SVG (Scalable Vector Graphics) format. Libraries like VexFlow, [8] Guido, [9] Verovio [10] or abcjs [11] can render music scores on the web. However, not all of them provide an API to interactively deal with the musical structure (model) behind the score.

The challenge we are facing is a need for a JavaScript-compatible musical model system that is able to:

1. Calculate and compose abstract music (like a MIDI file),
2. Play via WAM synthesizers,
3. Be displayed as a musical score.

Therefore, we created a JavaScript library called Sol for the calculation of different musical concepts. [12] It aims to deal with the musical model issue in the web-based CAC system. Then, a WebAssembly version of Guido [13] is used to render the score as it provides the AR (Abstract Representation) API to easily convert the musical model to the score. Finally, these feature are integrated as a package [14] into JSPatcher.

5.2 Musical Model

Musical notation is basically a representation of the musical structure based on a set of different musical concepts and a composition of instances of these concepts. To facilitate the CAC, the modeling of these concepts, or generally music theory, should be implemented in a digital way to allow the calculation between the musical concepts.

Yet, modern music theory is so complex and diverse that its modeling cannot be all-inclusive. The Sol library is only a proof of concept with a covering of very basic but necessary concepts in a common CAC workflow.

5.2.1 Pitch and Note

The difference between a pitch and a note can be ambiguous. In our library, "note" means different "pitch classes" in an octave. A note can be A, B, C, D, E, F and G with any number of sharp or flat accidentals.

[8] `https://www.vexflow.com/`
[9] `https://guido.grame.fr/`
[10] `https://www.verovio.org/`
[11] `https://www.abcjs.net/`
[12] Open-sourced on `https://github.com/fr0stbyter/sol`
[13] `https://github.com/grame-cncm/guidolib`
[14] `https://github.com/jspatcher/package-cac`

Text strings can be used to construct a Note object. For example, "C###" "Db" or "Bx" are recognizable as "note C with 3 sharps" "note D with 1 flat" and "note B with double sharps." Internally, the note name and the accidentals will be preserved and can be used for further calculation. Integer numbers can also be used to construct a note as the offset in semitones from note C, in this case, the note name and the accidentals will be determined automatically.

Pitch is an extended Note object with additional octave information. It is then a more concrete concept as the frequency can be calculated. It is also possible to get a pitch from a frequency with an approximation of a semitone. The pitch's offset follows the $C4 = 60$ standard.

Both Note and Pitch object supports some kinds of mathematical calculations. Adding or subtracting a number from a Pitch will change by semitones its offset. The offset difference can also be calculated between two Pitches For example, the following code creates a C4 pitch, by adding 12 to get a C5 pitch, then by subtracting a C4 pitch to get 12.

```
const pitch = new Pitch("C4");
pitch.add(12).toString(); // => "C5"
// Now pitch is C5
pitch.sub(new Pitch("C4")); // => 12
```

The multiplication between a pitch and a number is supported to calculate an approximated pitch based on the current pitch as the fundamental frequency and a multiplication ratio. The division is also possible between the pitch and a number or another pitch to calculate a new pitch or the frequency ratio between them. For example, the following code creates a C3 pitch, by multiplying 4 (to its frequency) to get a 2 octaves higher pitch C5, then by multiplying 1.5 to get a perfect fifth higher.

```
const pitch = new Pitch("C3");
pitch.mul(4).toString(); // => "C5"
// Now pitch is C5
pitch.mul(1.5).toString(); // => "G5"
```

5.2.2 Interval

Interval is the distance between two pitches, but it is more complex than just the number of semitones. With different note names and accidentals, the same distance, in equal temperament, can have different intervals such as diminished fourth and major third. Its calculation involves three properties: quality, degree and octave.

In Sol, an interval can be created with a text string, a frequency ratio, or from these properties. "A4" "d5-1" "P5" "M9" "m10+1" are usable strings for "augmented fourth" "diminished fifth with one octave lower" "perfect fifth" "major ninth (internally will be transformed to a major second with one octave higher)" "major tenth with one octave higher (internally a major third with two octaves higher)."

Here, the distance between two pitches can be "negative." In one octave, the interval between note F and note B is augmented fourth, and the interval between note B and note F will be "diminished fifth with one octave lower." The operation can be executed using the following code:

```
const b = new Note("B");
const f = new Note("F");
const i1 = f.getInterval(b);
i1.toString(); // => "A4"
const i2 = b.getInterval(f);
i2.toString(); // => "d5-1"
```

`i1` and `i2` from the code above are two intervals. They can be used for addition and subtraction from notes and pitches.

5.2.3 Chord

A chord is basically a set of pitches aligned vertically and executed at the same time. In our library, the Chord interface includes a base note or pitch and an array of intervals. It has methods such as inverse up and down, move up and down, etc. This model is chosen to easily move the chords, and also to solve the following question more easily:

"How to find the missing fundamental frequency of a given chord, if any?"

CAC is often used for spectral music composition algorithms, which involves the calculation of harmonics or the fundamental frequency. The *missing fundamental* of a sound [13] is an interesting topic in the scope. It has been demonstrated that the frequency of the pitch heard in response to a set of two or more successive harmonics corresponds to the greatest common divisor (GCD) of the harmonic set, even when there is no spectral energy at that frequency [14]. From a signal perspective, the autocorrelation algorithm can be used to find the periodicity of a sound and to determine the missing fundamental frequency. As we already have the frequencies from every pitch of the given chord, we can simply calculate the GCD of the frequencies.

However, the missing fundamental frequency should be an approximated value, as we are under the equal temperament, and human ears has a limited frequency discrimination capacity [15].

Therefore, we calculate a commonly approximated ratio between pitches from multiple ratios given by the intervals, we choose $\frac{1}{3}$ of a semitone, which is nearly a 2% difference, as a threshold of the frequency discrimination and the temperament compensation.

As a result, the algorithm is able to recognize that a dominant seventh chord has a ratio of $4:5:6:7$ and a missing fundamental on the two octaves lower dominant degree.

```
const chord = new Chord(
  new Pitch("C4"), new Pitch("E4"),
  new Pitch("G4"), new Pitch("Bb4")
);
chord.ratio; // => [4,5,6,7]
chord.phantomBase.toString(); // => "C2"
```

The code above calculates a dominant seventh chord on C4, it returns that its missing fundamental (`phantomBase`) is C2.

5.2.4 Duration

A musical value (duration) in fractions of beats can be expressed using the Duration object. Its constructor accepts a

fraction (numerator and denominator) or a text string such as “4n” “8nd” or “16nt” for “a quarter” “a dotted 8th” and “a triplet 16th.” Mathematical operations like compare, add, subtract, multiply and division are also available. For example, we create in the following code the length of a quarter note (one beat), then multiply it by 1.5 to get a dotted quarter, divide it by 9 to get a triplet 16th, finally add a quarter to it to get a length of $\frac{7}{24}$ beat.

```
const dur = new Duration(1, 4); // 1 beat
dur.mul(1.5); // 3/8 beat
dur.div(9); // 1/24 beat
dur.add(new Duration(1, 4)); // 7/24 beat
```

5.2.5 Sequence

Combining the presented models, a sequence can be formed with an array of chords or null values for a rest, and their duration. The sequence interface contains minimum data to render a score segment. It can also be compiled to a MIDI file with a time signature and BPM (beats per minute) information.

5.2.6 Others

Other musical concepts such as Velocity, Scale, Tonality, Track, Instrument, Genre, etc. are added to the library as well. However, these concepts are more complex and still need to be improved in future works.

5.3 Score Rendering

The Guido engine [16] is originally a C++ library aimed to compile plain text as GMN (Guido Music Notation) and to an SVG format file for the score. But it also has two intermediate steps, AR (Abstract Representation) and GR (Graphical Representation) that allow rendering a score from its API [17].

Through an Emscripten toolchain, the engine has been compiled to WebAssembly with most of its API and released as a JavaScript library at NPM (Node Packages Manager) online. To use the Sol library with the Guido API, we added a method `toGuidoAR` to the `Note`, `Pitch`, `Chord` and `Sequence` to perform function calls, construct the musical structure in the Guido AR.

In JSPatcher, the Guido engine runs in a separate thread using the Web Worker API. A `guido.view` object is available to accept a Guido AR instance and shows directly the compiled SVG score in the object UI.

5.4 Other CAC-related Objects

Manipulations of arrays (lists) of pitches, durations, velocities are common tasks in a CAC work. Like the numerous array-related functions in OpenMusic, we also need to provide them in JSPatcher.

Array versions of JavaScript operators are available as objects with the prefix “[]” like `[]+`, `[]*`, `[]&&`, which are array versions of “add” “multiply” and “logical and.” They accept an array as the first input. When the second input is an array, values in two arrays will be applied to the function respectively according to their index; otherwise, every value in the first array will be applied to the functions with the second value.

We also added some functions existing in OpenMusic like `x2dx` (calculate the difference between values in an array), `dx2x` (integrate an array), `permute`, `combinations`, `arithmSer` (create an arithmetic series), `fiboSer` (create a Fibonacci series), etc.

6. EXAMPLE

The following patcher (Figure 7) shows a CAC project combining the presented features.

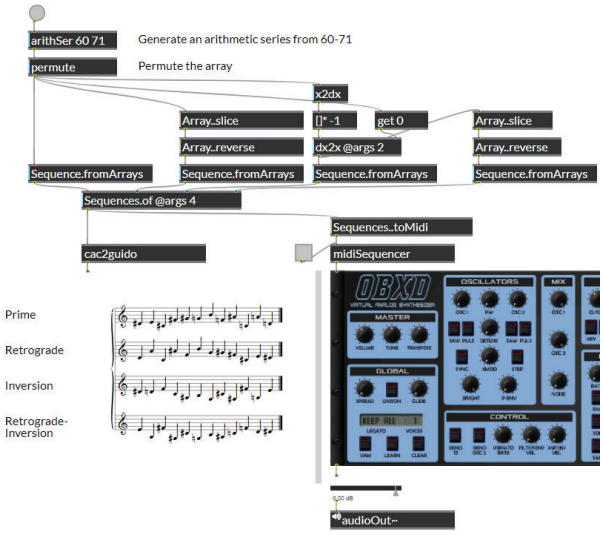


Figure 7. An example of CAC with a MIDI player

The first part of the patcher starts with a button that will create a Bang on click. The Bang generates an arithmetic series between 60 and 71, then randomly permutes the array to get the prime form of a dodecaphonic series. Its inversion is then calculated using the first value and the intervals between each value .The reversions of the two forms are calculated using a copy of the arrays. Using the Sequence and the Sequences functions, we now have a four-voice music segment. The score is displayed using the `cac2guido` object, and its MIDI file is generated and passed to a sequencer. Below the sequencer, a WAM synthesizer is loaded and waiting for real time MIDI messages. It produces audio data that is connected to the physical output. With the toggle, users can play or stop the sequence.

7. CONCLUSIONS

The scope of sound and music computing becomes wider today due to various possibilities to create music rapidly or even in real time. The development of machine learning and AI makes automatic composition possible and continuously inspires musicians to cooperate with the computer in the composition process. The accessibility offered by modern web technology with its highly active ecosystem is an opportunity for us to bring music programming and

CAC systems to a new stage. For instance, an artwork created on JSPatcher can be easily shared with people by designing a presentation mode of the patcher in which only selected boxes is showing with specific position and size, and specifying in the URI the location of the project file and the patcher filename, with additional options such as hiding the editor UI or make the patcher read-only. [15] In addition, UI components in patchers can also be interacted using touch devices which could make the design workflow simpler than existing VPLs on native platforms.

Today, audio processing in the web do have some limitations, especially the performance cost and the reliability compared to native C/C++ programs due to the implementation of different browsers, i.e. lack of optimization for multi-core CPUs. Even though, the web platform is still attractive to both developers and users. It has minimized the barrier for the deployment and use of any computer program from any device.

The described additions and improvements on JSPatcher are just the first steps towards a complete online IDE for sound and music computing. Feedbacks given by musicians from the community shows high interest in the future possibilities of JSPatcher such as supports for INScore, [16] collaborative editing and messaging via the network. As the project is open-sourced [17] with the examples and SDK (Software development kit) provided, contributions from third-party are feasible and welcome.

8. REFERENCES

[1] S. Ren, L. Pottier, and M. Buffa, "Build webaudio and javascript web applications using jspatcher: A web-based visual programming editor," in *Proceedings of the International Web Audio Conference*, ser. WAC '21, L. Joglar-Ongay, X. Serra, F. Font, P. Tovstogan, A. Stolfi, A. A. Correya, A. Ramires, D. Bogdanov, A. Faraldo, and X. Favory, Eds. Barcelona, Spain: UPF, July 2021.

[2] M. Puckette and D. e. a. Zicarelli, "Max/msp," *Cycling*, 1990.

[3] M. Puckette, "Pure Data," in *Proceedings of the International Computer Music Conference*. Thessaloniki, Hellas: The International Computer Music Association, Sep. 1997, pp. 224–227.

[4] J. Bresson, C. Agon, and G. Assayag, "Openmusic: Visual programming environment for music composition, analysis and research," in *Proceedings of the 19th ACM International Conference on Multimedia*, ser. MM '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 743–746.

[5] M. Puckette, "The patcher," in *Proceedings of the International Computer Music Conference*. San Francisco, United States: Computer Music Association, 1986, pp. 420–429.

[6] H. Choi, "Audioworklet: the future of web audio." in *Proceedings of the International Computer Music Conference*, Daegu, South Korea, Aug. 2018, pp. 110–116.

[7] Y. Orlarey, D. Fober, and S. Letz, "FAUST : an Efficient Functional Approach to DSP Programming," in *New Computational Paradigms for Computer Music*, E. D. France, Ed. Paris, France: Delatour, Jan. 2009, pp. 65–96.

[8] J. Vilk and E. D. Berger, "Doppio: Breaking the Browser Language Barrier," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2014, pp. 508–518.

[9] B. Powers, J. Vilk, and E. D. Berger, "Browsix: Bridging the Gap Between Unix and the Browser," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 253–266.

[10] J. Kleimola and O. Larkin, "Web audio modules," in *Proceedings of the Sound and Music Computing Conference*, Jul. 2015.

[11] M. Buffa, J. Lebrun, J. Kleimola, O. Larkin, and S. Letz, "Towards an open web audio plugin standard," in *Companion Proceedings of the The Web Conference 2018*, Lyon, France, Apr. 2018, pp. 759–766.

[12] M. Buffa, J. Lebrun, S. Ren, S. Letz, Y. Orlarey, R. Michon, and D. Fober, "Emerging w3c apis opened up commercial opportunities for computer music applications," in *The Web Conference 2020 DevTrack*, Taipei, Apr. 2020.

[13] A. Seebeck, "Beobachtungen über einige bedingungen der entstehung von tönen," *Annalen der Physik, 2nd ser.*, vol. 53, pp. 417–436, 1841.

[14] D. A. Schwartz and D. Purves, "Pitch is determined by naturally occurring periodic sounds," *Hearing Research*, vol. 194, no. 1, pp. 31–46, 2004.

[15] B. C. J. Moore, "Frequency difference limens for short-duration tones," *The Journal of the Acoustical Society of America*, vol. 54, no. 3, pp. 610–619, 1973.

[16] H. H. Hoos, K. Hamel, K. Renz, and J. Kilian, "The GUIDO notation format: A novel approach for adequately representing score-level music," in *Proceedings of the 1998 International Computer Music Conference, ICMC 1998, Ann Arbor, Michigan, USA, October 1-6, 1998*. Michigan Publishing, 1998.

[17] D. Fober, Y. Orlarey, and S. Letz, "Scores level composition based on the GUIDO music notation," in *Non-Cochlear Sound: Proceedings of the 38th International Computer Music Conference, ICMC 2012, Ljubljana, Slovenia, September 9-14, 2012*. Michigan Publishing, 2012.

[15] The URI for share could be like `https://fr0stbyter.github.io/jspatcher/dist/?projectZip=../../soundcraft/Soundcraft6.zip&file=020.jspat&runtime=1`

[16] `https://inscore.grame.fr/`

[17] `https://github.com/Fr0stbyteR/jspatcher`

A POLY-TEMPORAL PROGRAMMING ENVIRONMENT FOR LIVE SHOWS AND INTERACTIVE INSTALLATIONS

Martin Fouilleul
Sorbonne Université - STMS Ircam
martin.fouilleul@ircam.fr

Jean-Louis Giavitto
CNRS - STMS Ircam
jean-louis.giavitto@ircam.fr

Jean Bresson
Ableton, Berlin - STMS Ircam
jean.bresson@ircam.fr

ABSTRACT

In this paper we present *Quadrant*, a prototype programming environment for designing and performing temporal scenarios. Such scenarios can be used to trigger cues and control various parameters of live shows and interactive installations, such as audio and video playback, lights, or mechatronics.

Quadrant features a structure editor operating on a syntax tree that intermingles textual tokens and graphical user interface elements. This allows specifying scenarios algorithmically using a domain specific language, while expressing continuous time transformations with graphical curves.

Quadrant uses an imperative synchronous language to express concurrent poly-temporal scenarios. Scenarios are compiled on-the-fly into a bytecode that is run by a virtual machine. A temporal scheduler organizes the execution of concurrent parts of that bytecode along multiple abstract time axes, mapping abstract dates and delays of the program onto real time using a differential equation solver.

The virtual machine feeds back execution information to the structure editor, which reflects that information by highlighting executed statements or displaying progress wheels and status icons directly in the code. This allows an operator to easily monitor and pace the progression of the scenario.

1. INTRODUCTION

Timing and interactions are two critical aspects of almost any live show. Likewise, a large number of art installations are experienced within a specific temporality, and/or feature complex interactions. These two broad categories of artworks also increasingly involve technological artifacts and processes, as artists take advantage of them for creative purposes. Generative audio, shaders, video mapping, LED arrays, various sensors and robotic devices, are a few of the elements that can now participate in a rich network of temporal interactions with the performers and the audience.

In this context, technical designers or tech-savvy artists need tools to plan and control the *temporal scenario* of the show or installation. Such tools face somewhat opposed requirements: they must offer tight control over the scenario's timing, while remaining flexible and open enough to allow creative outcomes. They must provide a useful view of the show at multiple levels, e.g. from the broad strokes of the plot down to the details of a fader's automation. They must be ergonomic and easily controllable in live, while allowing the design and automation of complex processes. Tradeoffs are to be made, and can be made in a variety of ways. Furthermore, there is no one true solution that will work for all artists and every show. The design space is thus quite large and calls for exploration.

A number of tools have been proposed that enable the authoring and execution of temporal scenarios for various types of media. Cuelist softwares such as QLab [1], Medialon [2] or Smode [3] use nested lists of cues with ad-hoc timing behaviours, and/or timelines reminiscent of audio sequencers. Iannix [4] uses a spatial metaphor with 3D objects and trajectories to organize and schedule actions. Ossia Score [5] adopts a flow graph model which provides some degree of abstraction and programmability, while still conveying a spatial metaphor of time. Formula [6], Antescofo [7] or ChucK [8] tackle the problem by defining domain specific programming languages with temporal semantics. Gibber [9] relies on a general purpose scripting language (Javascript) and notation conventions to build live-coded multimedia performances.

In this work, we present a prototype environment called Quadrant, that aims at bridging the gap between a programming language approach and a more user interface focused point of view [1]. Quadrant consists of three components: a structure editor, a non-textual domain-specific language with a bytecode compiler, and a virtual machine. Bringing the full expressive power of a programming language to technical users affords them both precise control and unbounded extensibility. In a standard textual approach, this may come at the cost of learnability, efficiency, ergonomics and immediate feedback. However, the tight integration of Quadrant's components, and the structured representation of temporal scenarios they share, enables important user experience improvements.

2. QUADRANT STRUCTURE EDITOR

Quadrant features a structure editor named *Qed* to encode temporal scenarios. Structure editors let users view and edit structured data, using specific knowledge of the underlying format. This is to contrast with text editors, which

[1] A short video overview of Quadrant is available here: https://youtu.be/_wGPEDwp1AA.

Copyright: © 2022 Martin Fouilleul et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

operate on mostly unstructured streams of bytes, even when those streams are meant to encode some structure, such as a program. Although structure editors mostly gained wide adoption *outside* computer science academia [2], there is a long history of research into structural code editing. It ranges from early syntax-directed editors, such as MENTOR [10], the Cornell Program Synthesizer [11], or editor generators explored by the Gandalf Project [12], to more recent research rooted in type theory and functional programming such as Hazel [13] or Tylr [14], non-textual language generators like JetBrain's MPS [15], or attempts at designing general purpose structured formats such as Dion [16] or Infra [17].

2.1 Structured Data

The goal of Qed is to provide more ergonomic input and visualization models for temporal features of the language, as well as useful feedback about the execution of the program. However the editor should in some cases preserve the *feel* of text editing. First, it is still an appropriate *local* model for a lot of interactions (e.g. typing exact numeric values, simple arithmetic expressions, and obviously, text strings). More fundamentally, the editor should ease exploration of the user's problem space and incremental progress towards a solution. This sometimes implies relaxing structural requirements and allowing the user to navigate through error states. This departs from the insistence of most structure editors on making syntax or type errors impossible, which in our opinion is the main contributor to their perceived "stiffness" or lack of flexibility.

Qed is thus less structured than most structure editors, and defers checking some of the structural constraint to later stages of the compiler pipeline. The editor operates on a tree structure composed of *cells*. Cells can be simple tokens such as numbers or textual identifiers, or lists of other cells.

2.2 Navigation and Edition

The editor maintains a cursor which corresponds to a position inside the tree. The cursor can be described by a triple (p, r, o), where p is the parent cell under which the cursor lies, r is the children of p which is immediately on the right of the cursor (which can be null if the cursor is at the end of p's children list), and o is a text offset into the textual data of p, if any.

The cursor can be moved backwards or forwards in depth-first traversal order using left and right arrow keys, It can also be moved to the position that is *visually* upwards or downwards, using up and down arrow keys. The editor displays hints to help visualize the position in the tree: the parent cell is rendered against a light gray background, and the cells left and right to the cursor are indicated respectively by blue and green underlines.

A secondary cursor called the *point* is used to select parts of the tree. The point normally follows the movement of the cursor, unless the shift key is pressed, in which case it stays in place. To infer a selection from the cursor and point, we first determine their closest common ancestor $\mathscr{P}$. The selected range is then the minimal forest consisting of consecutive children of $\mathscr{P}$ which contains the point and the cursor. This allows growing and shrinking the selection to the next meaningful syntactic boundary as the cursor moves towards or away from the point.

Selections can be deleted, copied and pasted with standard shortcuts. In addition, when the cursor is not inside a string literal cell, the editor recognizes some keystrokes as special editing shortcuts to allow editing the structure of the tree: for instance, pressing the left parenthesis `(` key creates a simple list cell, while the shortcut `Cmd+(` encloses the cell directly right to the cursor into a new list cell.

Otherwise, entering text replaces the current textual selection by the input characters, much as in a standard text editor. Once the input characters have been inserted, the cell's text is tokenized, which may modify the current cell or split it into several new cells.

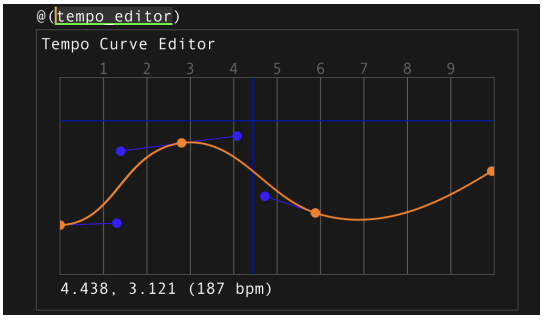


Figure 1. Quadrant Tempo Curve Editor

2.3 Tempo Curve Editor

Since the editor is aware of language structure, it can recognize certain syntactical constructs, and attach custom graphical user interface elements to them. This is the case with the tempo curve editor (Figure 1), which allows specifying the tempo of a block of code with Bézier curves, whose control points can be adjusted with the mouse. This offers both an easier input model and a better visualization of continuous curves than a purely textual construct.

The user can zoom and scroll in an infinite grid with adaptive markings. Clicking on the curve splits the Bézier curve under the mouse. Alt-clicking on an endpoint deletes it and merges the two pieces on each side. The user can drag control points with the mouse to shape the curve as needed. A dark blue crosshair and textual coordinates help precisely control the position of the mouse pointer. Clicking an endpoint and hitting `Ctrl+C` toggles the tempo continuity constraint at this point. When the constraint is disabled, the last and first points of two consecutive Bézier curves can be dragged up and down independently.

3. TEMPORAL MODEL

We briefly discuss the main aspects of Quadrant's temporal model here. For a more detailed explanation, we refer the

[2] Word processors, cell sheets or 3D modelling softwares are good examples of widely used structure editors. Meanwhile in 2022, computer science papers, including this one, are still being written in LateX.

reader to a previous work [18] on poly-temporal scheduling with parametric tempo curves.

3.1 Symbolic timeframes

A musical score ascribes temporality to musical events using symbolic dates and durations (e.g. beats and notes values). Symbolic time is mapped to performance time by the interpreter, following tempo indications, cultural conventions, and interpretative choices. In the realm of computer music environments, several authors have proposed extending this mapping recursively, creating a hierarchy of symbolic timeframes, where a timeframe is related to its parent by a time transformation [6, 19, 20]. Time transformations can be represented and applied by a variety of techniques: precomputed time maps [19, 21], tempo and time shift maps [22], beta functions [23], piecewise tempo curves built on tweening functions [24].

Mapping a date from one timeframe to another using a tempo curve implies solving a differential equation [18]. Instead of relying on predefined functions with known integrals, Quadrant uses a numerical solver to integrate parametric tempo curves. In particular, we chose to construct tempo functions from Bézier curve pieces, which are more versatile than standard tweeners and allow easy tweaking of control points by a user through a graphical interface.

3.2 Scheduling

Quadrant's poly-temporal model relies on concurrent tasks (implemented as stackful coroutines, or *fibers*) managed by a cooperative scheduler. Each task represents a sequence of interleaved computations and delays happening in a given timeframe.

Computations are predictably ordered and are considered to happen instantaneously with respect to symbolic time. This makes Quadrant's model similar to that of synchronous languages [25], with a few differences that we detail below.

Strictly speaking, most synchronous languages don't have an inbuilt notion of time, and can only react to signals. This does not pose theoretical difficulties but does make some scenarios cumbersome, since one must rely on introducing and counting external "clock" signals. This downside is discussed in [26], which also proposes extending the host context of Esterel to allow a program to schedule its own wakeup time when returning from its `step` function. We use a somewhat similar approach in Quadrant, where tasks can pause for a requested amount of symbolic time.

We allow temporarily removing some task from the synchronous scheduling mechanism to have them executed in a background task pool. This permits graceful handling of blocking or asynchronous operations, such as input/output, without stalling the scheduler.

Finally, while most synchronous languages are concerned with providing hard real-time guarantees, we are mostly interested in providing a predictable yet flexible concurrency model, and only consider soft real-time goals on a best-effort basis.

4. QSCORE LANGUAGE

Quadrant provides a custom language named QScore to encode temporal scenarios. As discussed in section 2, it is a non-textual language: although its source representation is mostly *displayed* as text, it is in fact a tree of cells that holds tokenized data or user interface widgets.

Since the tree structure is rendered explicit by the editor, in the form of indentations and parenthesis and through the use of s-expressions, the appearance of the source is quite reminiscent of the Lisp programming language. This is however the end of the similarity, since QScore has very different characteristics than typical Lisp dialects. Indeed, QScore is an imperative, statically typed language with (mostly) unmanaged memory. It has built-in cooperative concurrency and temporal primitives based on tasks. It is compiled to a bytecode run by a virtual machine. Since the language is statically typed, values are unboxed and memory layouts are known ahead of time, which means that the virtual machine can be fairly lightweight.

4.1 QScore Basic Constructs

We give here a brief overview of the language. Some more advanced features are shown in a short video here: `https://youtu.be/AmO9hczGkYU`.

4.1.1 Variables and Expressions

The following form declares a variable `name` with type `typeSpec` is the current scope, and initializes it with the expression `initExpr`:

```
(var name typeSpec initExpr)
```

A variable can be assigned a new value of its type using the `set` form:

```
(set name val)
```

Common operators can be used in prefix notation to compute numeric or boolean expressions, such as

```
(and foo (< bar (* 3.2 baz)))
```

4.1.2 Named Types

A named type can be defined by associating a name to a type specification, using the form:

```
(type typeName typeSpec)
```

where `typeName` is an identifier and `typeSpec` is a type specification. QScore predefines a number of primitive named types for signed and unsigned sized integers, floating point numbers, booleans and a `void` type.

4.1.3 Arrays and Slices

An array is fixed-size container of contiguous elements of the same type. An array type is specified using this form:

```
(array count typeSpec)
```

A slice is a reference to a contiguous range of elements of an array. Its number of elements need not be known at compile time. Its type is specified using this form:

```
(slice typeSpec)
```

4.1.4 Structures

A structure is a collection of named fields, each with their own type. A struct type is specified using the form

```
(struct name1 typeSpec1
        name2 typeSpec2
        ...)
```

4.1.5 Pointers

A pointer is an address of a value of a given type. A pointer type is specified using the form:

```
(ptr typeSpec)
```

The `ref` form takes the address of an addressable operand. The `unref` form allows accessing the underlying value:

```
(ref addressableOperand)
(unref pointer)
```

4.1.6 Control Flow

QScore has the usual basic control flow constructs like conditionals and loops, and lexical scoping:

```
(if condition
    branchIfTrue
    branchIfFalse)

(for initStatement
     conditionExpression
     iterationExpression
   body)

(while condition
   body)

(do
   body)
```

4.1.7 Procedures

Procedures can be defined using the `def` form:

```
(def procName (param1 typeSpec1
               param2 typeSpec2
               ...
               -> returnTypeSpec)
   body)
```

A procedure returns values using the `return` form:

```
(return val)
```

Procedure calls use prefix notation:

```
(procName arg1 arg2 ...)
```

4.2 Temporal Features

4.2.1 Pause

A task can request to be paused for a given duration and yield to the scheduler using the `pause` instruction:

```
(pause duration)
```

where `duration` is a numeric value specifying the duration of the pause in the timeframe of the current task.

4.2.2 Standby

The `(standby)` instruction suspends the execution of the current task until it is resumed by a trigger action sent from the editor.

4.2.3 Flow and Futures

The `flow` form launches a new task to execute its body, and yields to the scheduler.

```
(flow
   body)
```

A `flow` block can refer to variables from its outer scopes. In this case, it captures those variables, and the activation frames in which these variables live are kept valid at least until the block ends or returns.

The `@(tempo_editor)` attribute can be used right after the `flow` keyword to attach a tempo curve editor to the block (see subsection 2.3). The tempo curve is then used to map the timeframe of the task running the `flow` body to and from its parent task's timeframe.

A `flow` block may contain `return` statements, which must all be of the same type. Otherwise it is considered to return `void` after the last statement. In the calling task, the `flow` form evaluates to a *future*, which is a typed handle to the new task concurrently executing the `flow` body. The type specification of a future is

```
(future typeSpec)
```

where `typeSpec` is the specification of the type returned by the `flow` block.

The `wait` form suspends the current task until a given `flow` block returns. The task is then resumed and the `wait` form evaluates to the value returned by that block.

```
(wait someFuture)
```

The `timeout` form suspends the current task until a given `flow` block returns, or a given local timeout expires, whichever is the earliest. It evaluates to a boolean value which indicates if the `flow` block has returned.

```
(timeout someFuture maxDuration)
```

`wait` and `timeout` forms can use the `@(recursive)` attribute to recursively wait for a `flow` block and all the `flow` blocks it launched to have returned.

Tasks are reference-counted to keep the associated data (in particular, the returned value) alive for subsequent waits and timeouts. It is the responsibility of the programmer to manifest their intent in this regard by using the `fdup` and `fdrop` forms, which respectively duplicate a future and increment the reference count of the underlying task, or drop a future, which decrements this reference count.

```
(fdup someFuture)
(fdrop someFuture)
```

4.3 Compiler pipeline

The cell tree is processed by several pipelined stages before it can be executed by the virtual machine (Figure 2).

It is first traversed by a parser to produce an abstract syntax tree. A checker then traverses the syntax tree to create a typed syntax tree and symbol tables. The parser and the checker also produce an error log, with each error attached to the range of cells from which it originates. This error

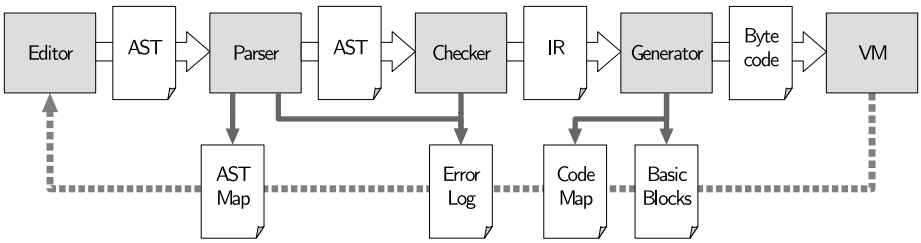


Figure 2. Quadrant Pipeline

log is used by the editor to draw error underlines and display a pop-up panel with error messages when the cursor is on a faulty cell.

When the program is valid, a generator uses the checker output to generate a serialized representation which can be loaded and executed by the virtual machine. This representation contains directives to setup the address space of the program, initialize static data, and load foreign libraries and symbols, as well as the bytecode corresponding to each procedure of the program.

The compiler pipeline generates several debug tables:

- An AST map, which maps cells to AST nodes. This is used by the editor to query syntactic properties of cells, in order to perform auto-layout and syntax highlighting.
- A code map, which contains mappings between cells and bytecode offsets and vice-versa. This table is used by the editor to translate source locations to and from bytecode locations when communicating with the virtual machine.
- A block map, which maps bytecode offsets to basic execution blocks. Basic execution blocks are used by the editor to highlight portions of the source when they are executed by the virtual machine.

5. QUADRANT VIRTUAL MACHINE

The virtual machine of Quadrant is a fairly simple stack machine whose design draws some inspiration from the Quake III Arena VM by Id Software [3] .

5.1 Instructions Encoding

Code is segregated from program data. Instructions consists of a one byte opcode, followed by zero to three immediate operands. The size of each operand is encoded in the opcode and can be 1 to 8 bytes. Along with standard operations such as loads and stores, arithmetics, comparisons, logic, conversions, and jumps, the instruction set includes specialized opcodes to manage tasks and control time flow.

5.2 Address Space Layout

Tasks share the same linear address space, which is reserved through the operating system's virtual memory API at load-time and committed as needed on a page-by-page basis. Data loads and stores are confined to this fixed, contiguous region of memory, which is divided in several sections:

- `rodata`: this section is initialized at VM load-time with the static program data generated by the compiler pipeline, such as string literals and tempo curve descriptors.
- `bss`: this section is initialized to zero at VM load-time, and holds the program's global variables.
- `stack pool`: this uninitialized section is used to allocate fixed-size stacks for the tasks, using a pool allocator.
- `heap`: this uninitialized section is used to allocate objects of different sizes and lifetimes using a general purpose allocator.

5.3 Task Structure

The VM keeps track of tasks with a list of `vm_task` structures allocated from a dedicated pool. A generational index maps `future` values to `vm_task` structures. A `vm_task` structure holds the task's scheduling data and registers. It also has a slot to hold the return value of the task. If the task returns a structure, this slot holds a pointer to the return value, which is allocated on the heap.

In addition to the task structure, the VM also allocates a fixed size block from the `stack pool` for each task. This block is further split into two stacks: an operand stack, and a control stack.

For each task the VM maintains two reference counts: the number of `future` objects used to reference that task, as counted by `fdup` and `fdrop` forms, and the number of active captures of that task's control stack. When the number of captures drops to zero, the task's stacks can be recycled by the stack pool. When the active `future` count drops to zero, the memory allocated to hold the return value (if any) can be released to the heap. When both counts drop to zero, the VM can recycle the task structure.

5.4 Registers and Stacks

Each task has the following set of registers:

- `ip` (instruction pointer): this register points to the next instruction to execute.
- `osp` (operand stack pointer): this register points to the top of the operand stack.

[3] `https://github.com/id-Software/Quake-III-Arena/blob/master/code/qcommon/vm_interpreted.c`

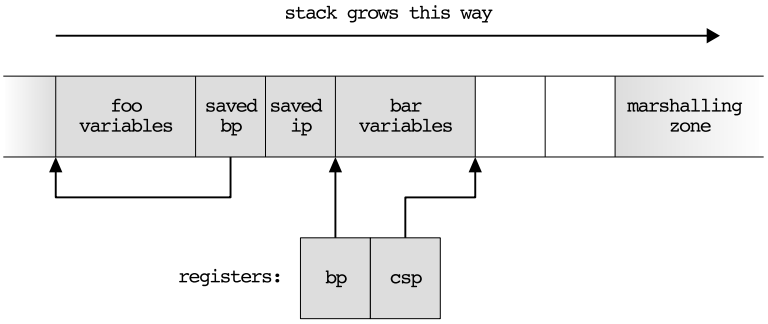


Figure 3. Control Stack Layout (here current procedure `bar` has been called by `foo`)

- `csp` (control stack pointer): this register points to the top of the control stack.
- `bp` (frame base pointer): this register points to the base of the current activation frame.
- `sr` (status register): this register holds status flags used by comparison and conditional jumps.

The operand stack consists of 8-byte aligned operands. Instructions that push or pop values of smaller size respectively zero-extend or mask those values.

The control stack consists of activation frames that contain the local variables of the task's active procedures (Figure 3). After the local variables, two 8-byte slots are reserved to store the `bp` and `ip` registers when calling a procedure. The *marshalling zone* after these slots corresponds to the location of the local variables in the next activation frame, and is used for passing arguments to procedures and new tasks.

5.5 Calling conventions

5.5.1 Regular Procedure Calls

When calling a procedure, the generator outputs opcodes to evaluate arguments on the operand stack, and then move them to the marshalling zone. If the procedure returns a structure, a return pointer is generated and passed as a hidden first argument. It then outputs a `call` opcode. This instruction copies the current frame base pointer and the instruction pointer to their respective slots at the end of the frame, then jumps to the address of the procedure. The procedure's code starts with a `enter` opcode that adjusts the `bp` and `csp` registers to the new activation frame.

5.5.2 Task Calls

A new task is created either for a `flow` block or for a regular procedure whose frame is captured by `flow` blocks. If necessary, the generator inserts an opcode to allocate memory for the return value prior to the call. The arguments are evaluated and collected to the marshalling zone the same way as a normal call. The `task` opcode then creates a child task and copies the arguments from the marshalling zone to the control stack of the new task. It then creates a `future` value for the new task and puts it on the caller's operand stack. The caller then yields to the scheduler to pass execution to the new task.

5.5.3 Returns

Upon return, the `csp` register is reset to the `bp` register. If the `csp` register then points to the base of the control stack, the VM pops the return value from the operand stack, stores it in the task's structure return field, and terminates the task. Otherwise, the previous `bp` and `ip` registers are popped from the control stack, which will return to the caller with the return value still on top of the operand stack.

5.5.4 Foreign Calls

The generator associates an index to each foreign procedure, and generates a listing of foreign dependencies and symbols, including type information describing each procedure's interface. At load time, the VM loads foreign libraries and symbols, and prepares the data structures used by the underlying FFI library (`libffi` [4]) into an array.

A foreign call start by the same argument evaluation sequence as regular calls. The `ffi_call` opcode then populates an argument buffer with pointers to the arguments inside the marshalling zone, obtains the FFI data using the procedure index, and uses the FFI API to make the call.

6. EXECUTION MONITORING AND PACING

The execution of a Quadrant program can be monitored and paced from within the editor (Figure 4):

- The editor highlights blocks of code in green as they are being executed.
- The editor shows a progress wheel when a task is paused. The proportion of the green arc shows the proportion of time elapsed with respect to the pause duration.
- The editor displays a spinning wheel composed of two green arcs when a task is waiting on a future.
- The editor shows a standby icon composed of two wavering green rectangles when a task is on standby.
- The editor can send triggers to resume a task suspended by a `standby` instruction

This is achieved by running the virtual machine and the editor in separate threads and have them communicate via message passing using a pair of wait-free ring buffers.

[4] `https://github.com/libffi/libffi`

```
(var f (future void) (flow
                       (pause 10)))
(flow
   (for (var i i32 0)
        (< i 10)
        (set i (+ i 1))
      (pause 1)))
(flow
   (wait f))
(standby)
```

Figure 4. Quadrant Execution Monitoring

6.1 Execution Blocks

An execution blocks is a contiguous, uninterrupted block of bytecode (i.e. containing no jumps and yields) that maps to a contiguous range of cells. The generator produces a table pairing the start offsets of execution blocks to the range of cells they originate from.

When executing a jump or a yielding instruction, the VM sends a `trace` message to the editor, containing the target offset of the jump. The editor then uses the blocks table to map it to a range of executed cells, and displays a flashing green background behind those cells.

Execution may fall back from one execution block to another without a jump or yielding instruction. This is e.g. the case when joining from the `false` branch of an `if` form to the next execution block. It can also happen when the generator reorders instructions compared to how they appear in the source. In these cases, the generator inserts a `trace` opcode at the beginning of the second execution block, which explicitly instructs the VM to emit a `trace` message.

6.2 Progress Reports

The scheduler runs a special fiber at a fixed frequency to monitor the progress of the scenario. For each task, the monitoring fiber collects the status of the task, the time remaining, and a call stack consisting of the bytecode offset of all call-sites for that task up to the current `ip`. It then sends a `progress` message to the editor containing that information. The editor then uses the code map to display progress wheels, spinning wheels or standby icons next to all call sites of a suspended task.

6.3 Standby Triggers

When the cursor is on a `standby` form, and the user hits the `Ctrl+Space` shortcut, the editor uses the code map to send a `trigger` message with the bytecode offset corresponding to that form to the VM. The VM then resumes every task that is on standby at that particular location. This provides a way to manually pace the progress of the scenario from within the editor.

7. CONCLUSION

In this paper, we pointed out the increasing complexity of human and technological temporal interactions in live shows and art installations. After mentionning a several show control softwares and their paradigms, we introduced Quadrant, a new programming environment for designing and performing poly-temporal scenarios (section 1).

We discussed the motivations behind Quadrant's editor Qed, and presented its main features (section 2). We then explained the poly-temporal model of Quadrant (section 3), and gave an overview of QScore, its non-textual language with built-in temporal constructs (section 4). We also presented the architecture of Quadrant's virtual machine (section 5). We finally described how the Quadrant's editor and virtual machine cooperate to enable live monitoring and control of a temporal scenarios' performance (section 6).

7.1 Future Work

Quadrant is obviously still in its infancy. Its ability to interact, both with external devices and with a human operator, is thus quite limited. An immediate avenue for improvement is to extend interactivity on these two fronts.

The first one should be quite easily addressed by constituting a standard library of connectivity protocols, which would include e.g. UDP, TCP and WebSocket, as well as messaging protocols such as OSC, MIDI, or DMX. The foreign system of QScore should prove useful in this endeavour, as a way of leveraging existing code.

The user interactivity aspect is a little more challenging. Our plan is to implement breakpoints and stepping features similar to those of debuggers, and to build interactive workflows on top of it. That would include specialized tasks that act like random access cue lists or timelines, and mechanisms to control their execution from the editor. We would also like to consider live-coding. This could be implemented by appending basic execution blocks to the end of the code section, and patching old blocks with jumps to reroute control flow through these blocks at run-time.

Finally, an important problem to tackle is the automatic synchronization of QScore tasks to external processes, such as sequencers, score followers, or other Quadrant instances. Using the connectivity protocols evoked above, Quadrant could receive beat messages and generate tempo curves on the fly to catch up with these synchronization sources. This would in turn potentially open the way for performances using distributed and collaborative scores.

Acknowledgments

We would like to thank Allen Webster and Ryan Fleury of Dion Systems for sharing their perspective on structured editing and data formats, and being inspiring interlocutors.

References

[1] *QLab*. [Online]. Available: `https://qlab.app/`.

[2] *Medialon - Powerful Show Control Solutions*. [Online]. Available: `https://medialon.com/`.

[3] *Real-time compositing, media server and XR - SMODE*. [Online]. Available: https://smode.fr/.

[4] T. Coduys and G. Ferry, "Iannix. Aesthetical/symbolic visualisations for hypermedia composition," in *Sound and Music Computing Conference (SMC)*, 2004.

[5] J.-M. Celerier, P. Baltazar, C. Bossut, N. Vuaille, J.-M. Couturier, and M. Desainte-Catherine, "OSSIA: Towards a unified interface for scoring time and interaction," in *TENOR 2015 - First International Conference on Technologies for Music Notation and Representation*, 2015.

[6] D. P. Anderson and R. Kuivila, "A system for computer music performance," *ACM Transactions on Computer Systems*, vol. 8, no. 1, 1990.

[7] J. Echeveste, J.-L. Giavitto, and A. Cont, "A Dynamic Timed-Language for Computer-Human Musical Interaction," INRIA, Tech. Rep., 2013.

[8] G. Wang, P. R. Cook, and S. Salazar, "ChucK: A Strongly Timed Computer Music Language," *Computer Music Journal*, vol. 39, no. 4, 2015.

[9] C. Roberts, M. Wright, J. Kuchera-Morin, and T. Höllerer, "Gibber: Abstractions for Creative Multimedia Programming," in *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014.

[10] V. Donzeau-Gouge, G. Huet, G. Kahn, and B. Lang, "Programming Environments Based on Structured Editors: The MENTOR Experience," Tech. Rep., 1980.

[11] T. Teitelbaum and T. Reps, "The Cornell program synthesizer: A syntax-directed programming environment," *Communications of the ACM*, vol. 24, no. 9, 1981.

[12] A. N. Habermann and D. Notkin, "Gandalf: Software development environments," *IEEE Transactions on Software Engineering*, vol. SE-12, no. 12, 1986.

[13] C. Omar, I. Voysey, M. Hilton, J. Aldrich, and M. A. Hammer, "Hazelnut: A Bidirectionally Typed Structure Editor Calculus," *ACM SIGPLAN Notices*, vol. 52, no. 1, 2017.

[14] D. Moon, *Tylr*, hazelgrove, 2022. [Online]. Available: https://github.com/hazelgrove/tylr.

[15] *MPS: The Domain-Specific Language Creator by JetBrains*. [Online]. Available: https://www.jetbrains.com/mps/.

[16] *Dion Systems*. [Online]. Available: https://dion.systems/.

[17] C. Hall, T. Standley, and T. Hollerer, "Infra: Structure All the Way Down," *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, 2017.

[18] M. Fouilleul, J. Bresson, and J.-L. Giavitto, "A Polytemporal Model for Musical Scheduling," in *15th International Symposium on Computer Music Multidisciplinary Research*, 2021.

[19] D. Jaffe, "Ensemble timing in computer music," *Computer Music Journal*, vol. 9, no. 4, 1985.

[20] J.-L. Giavitto, J.-M. Echeveste, A. Cont, and P. Cuvillier, "Time, timelines and temporal scopes in the antescofo DSL v1.0," in *International Computer Music Conference (ICMC)*, ICMA, 2017.

[21] J. Rogers, J. Rockstroh, and P. N. Batstone, "Music-Time and Clock-Time Similarities under Tempo Changes," in *International Computer Music Conference*, 1980.

[22] H. Honing, "From Time to Time: The Representation of Timing and Tempo," *Computer Music Journal*, vol. 25, no. 3, 2001.

[23] J. MacCallum and A. Schmeder, "Timewarp: A graphical tool for the control of polyphonic smoothly varying tempos," *International Computer Music Conference, ICMC 2010*, 2010.

[24] *Curve - AntescofoDoc*. [Online]. Available: https://antescofo-doc.ircam.fr/Reference/compound_curve/.

[25] N. Halbwachs, *Synchronous Programming of Reactive Systems*. Kluwer Academic Publishers, 1993.

[26] R. von Hanxleden, T. Bourke, and A. Girault, "Real-time ticks for synchronous programming," in *2017 Forum on Specification and Design Languages (FDL)*, 2017.

MEASURING VIRTUAL AUDIENCES WITH THE MUSICLAB APP: PROOF OF CONCEPT

Dana Swarbrick[1,2] danasw@uio.no
Finn Upham[1,2] finnu@uio.no
Çağrı Erdem[1,2] cagrie@uio.no
Alexander Refsum Jensenius[1,2] alexanje@uio.no
Jonna Katariina Vuoskoski[1,2,3] jonnakv@uio.no

[1]RITMO Centre for Interdisciplinary Studies in Rhythm, Time, and Motion
[2]Department of Musicology, University of Oslo
[3]Department of Psychology, University of Oslo

ABSTRACT

We present a proof of concept by using the mobile application MusicLab to measure motion during a livestreamed concert and examining its relation to musical features. With the MusicLab App, participants' own smartphones' inertial measurement unit (IMU) sensors can be leveraged to record their motion and their subjective experiences collected through survey responses. The MusicLab Lockdown Rave was an Algorave (live-coded dance music) livestreamed concert featuring prolific performers Renick Bell and Khoparzi. They livestreamed for an international audience who wore their smartphones with the MusicLab App while they listened/danced to the performances. From their acceleration, we computed quantity of motion and compared it to musical features that have previously been associated with music-related motion, namely pulse clarity and low and high spectral flux. By encountering challenges and implementing improvements, the MusicLab App has become a useful tool for researching music-related motion.

1. INTRODUCTION

Virtual concert popularity increased as a result of the COVID-19 pandemic as musicians adapted to the social distancing requirements, and they livestreamed or recorded their performances for their audiences. However, how did this impact people's movement to music? It is generally known that music makes us move. Certain genres such as dance music make us move more even when we try to stand still [1]. Musical features, such as the "drop" in electronic dance music [2], or pulse clarity and low frequency spectral flux in popular music [3], evoke more movement from listeners. Furthermore, personal characteristics such as emotion [4], personality [5]–[7], and fan-status [8], and environmental factors such as social context [7] and liveness [8] influence music-related motion.

Movement is an important component of engagement with music because it not only influences the way that we enjoy music [9], but also the way that music is perceived [10]. While conventional music cognition may consider action and perception as distinct, research from neuroscience, psychology and behaviour indicate that perception and action are tightly linked. For example, when participants moved to a rhythm with an ambiguous meter every two or three beats, they perceived the meter as being duple or triple, respectively [11]. Movement can assist with musical timing perception [12]. Even when participants are instructed to simply imagine accented beats, this imagined meter is encoded in their brain signals as it would be if they had really heard the meter [13]. Embodied music cognition unites action and perception in a framework that situates the body at the centre of musical interaction [14], [15].

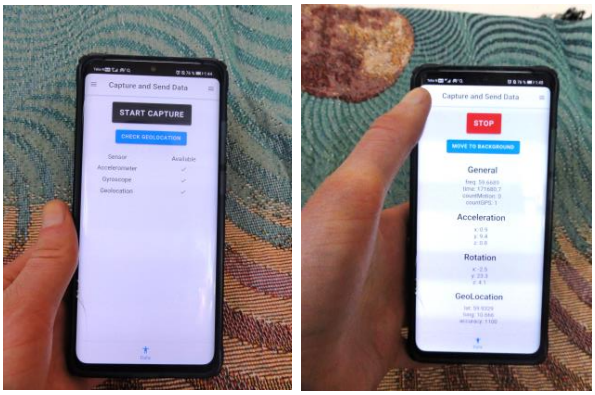


Figure 1: The MusicLab App is a mobile application that can be installed on participants' own smartphones.

Another important component of musical experience is its social nature. Music is a social phenomenon because performing and listening typically occur in the presence of others [16]. Even solitary music listening can be considered social because there is an imagined presence of the musicians. A concert is typically a very social experience in which artists and audience share time and space. A virtual concert may be viewed as a livestream (shared time) or pre-recorded. Livestreamed concerts are unique social contexts where audience members may be listening physically alone, but interacting online with other audience members and experiencing the concert at the same time. People may move differently depending on whether they are with others (e.g. move more as they dancing together or move less to fit in with social norms), or if they are alone (e.g. move more due to shyness or move less because they don't have others to communicate with). Embodied music cognition can support research into the social elements of music cognition as long as the social nature of

Copyright: © 2022 Swarbrick et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any médium, provided the original author and source are credited.

music is also considered when designing and interpreting research findings [17].

Motion can be measured with a variety of tools ranging in features including cost, portability, and precision. Optical motion capture systems record motion in 3 dimensions using cameras that emit and capture infrared light from reflective markers attached to participants. They are costly and often require a dedicated lab space, however the data has very high precision [18]. Video can measure motion for relatively low cost and with excellent portability, however it lacks the high-precision offered by the state-of-the art optical motion capture systems. Inertial measurement unit (IMU) sensors consist of an accelerometer, gyroscope, and magnetometer which measure acceleration, orientation, and geolocation, respectively. They are portable (present in most smartphones and can also be purchased separately) and are low-cost. Position measured from geolocation is low precision compared to video and optical motion capture systems, however IMU sensors measure acceleration at a high precision. Data privacy concerns are greater with video, audio, and motion capture data as IMU sensors do not record pictures of faces. Nonetheless, geolocation could reveal a participant's identity which is why there are several protocols for obscuring location data prior to publication [19]. A comparison between inertial sensors, video, and optical motion capture in a single group dance experiment highlighted that each measurement technique has its own strengths and weaknesses and combining systems is helpful for understanding motion [20].

The MusicLab App is a smartphone application available for both Android and Apple operating systems that collects motion from participants' own smartphone IMU sensors. The application can also collect survey responses through a connection with the University of Oslo's webform system. In accordance with current privacy regulations (specifically, the GDPR), both motion and survey responses are stored in secure servers at the University of Oslo (using the Nettskjema service). The app's source code is freely available and is in continuous development[1] . It was developed to be used in conjunction with MusicLab events, which feature a combination of a musical performance, a research component, a panel discussion, and "DJ"ing in the form of "Data Jockeying", in which a researcher demonstrates how the data that was collected will be analysed. These events aim to embody the principles of open science and FAIR data while managing copyright and privacy concerns [21].

In the summer of 2020, live-coding performers Renick Bell and Khoparzi livestreamed Algorave (live-coded dance) music for an international virtual audience[2]. The live-coded music was improvised and varied in pulse clarity with some sections being more danceable with a clear beat and others having a less clear pulse. Participants used the MusicLab App to measure their motion and respond to questionnaires. Motion and musical features were extracted and compared to understand their relations in a virtual concert. All music, code, and anonymized data was published under a CC-by 4.0 license and can be accessed at the OSF repository[3]. The main aim of this study was to provide a proof of concept for using the MusicLab App at a livestreamed concert.

2. METHODS

Participants were recruited and the concert was promoted through targeted Facebook advertising and performers' social networks.

2.1 Technical Set-up

The event host was situated in Finland so she joined a Zoom video call that was livestreamed into YouTube by the first author in Norway via Open Broadcaster Software (OBS). The YouTube stream key was sent to the first performer who livestreamed from Japan and then to the second performer who livestreamed from India. After the performances, the performers and researchers gathered on Zoom for a panel discussion. Following the panel discussion, a preliminary data analysis was live-coded to reveal to the audience how their data can be used (code: https://github.com/fourMs/MusicLab5).

2.2 Participant Instructions

Participants were instructed to download the MusicLab app provided in links in the video description. Participants provided consent in the MusicLab app in accordance with the Declaration of Helsinki, 2013. The Norwegian Centre for Research Data approved the study (741882).

Moderators provided technical support by responding to questions in the YouTube live chat. Participants were encouraged to place their phones on their upper bodies (such as in a shirt chest pocket) and to stand for the duration of the concert. They were told to dance if they felt the desire to do so.

Before the concert, participants responded to a survey in the MusicLab App that collected demographic information (age, gender, nationality), musical information (musician status, experience with Algorave and livestreams, fan-status), sensorimotor and social components of the Barcelona Musical Reward Questionnaire [22], the ten-item personality inventory [23], and the empathic concern and fantasy subscales of the interpersonal reactivity index [24]. After each performance they responded to questions on the performance (its danceability, familiarity, enjoyment, audio and video quality), motion (phone location, amount of their own motion to the beat and the performer's motion to the beat), social experience (connectedness to the performer and other audience members, perceived performer interaction with the audience, and whether there was anyone watching the performance with them), and their personal state (level of attention and amount of standing or sitting).

[1] https://github.com/fourMs/MoMoCapture

[2] Watch their performance here: https://youtu.be/hJ73IGYawuM

[3] OSF Repository DOI: 10.17605/OSF.IO/7J2GA

2.3 Alignment

To enable aligning the motion signals with the music, participants were instructed to begin recording while their phone was on a flat surface and to shake their phones twice when they heard the first sound from each performer. Preliminary testing indicated that this shake would be easily identifiable in the motion signal.

The accelerometer data was visually inspected during the two minutes that followed the start of the first sound from each performer. The first indication of a shake was presumed to be the shake. (There were a few participants that had several shake-like features in close proximity to the start of the song.) Since participants picked up their phones and then shook them at the first sound they heard, the initiation of the shake was manually selected as the start of the performance. Performance duration was used to demarcate the end of the performance.

2.4 Motion Features

Acceleration had average sampling rates near 60 Hz. However, this was variable between devices and also within some devices.

Acceleration signals were low-passed filtered with a Butterworth filter with a cutoff at 5 Hz to focus on meaningful human motion. The filtered acceleration was then downsampled to 10 Hz. The quantity of motion time series (QoM) was calculated as in equation (1) for the duration of the performance, taking the absolute change of the accelerometer data per sample to represent when participants were moving to the music [25].

$$\boldsymbol{QoM} = \sqrt{(\boldsymbol{x_{t2}} - \boldsymbol{x_{t1}})^2 + (\boldsymbol{y_{t2}} - \boldsymbol{y_{t1}})^2 + (\boldsymbol{z_{t2}} - \boldsymbol{z_{t1}})^2} \quad (1)$$

The resulting QoM data was smoothed using a Savitzky-Golay filter (order: 1, window: 299).

2.5 Musical Features

Musical recordings were trimmed in REAPER to remove silence from the beginning and end. Musical features were extracted using the MIRtoolbox for MATLAB (2019) [26]. We chose to analyze only musical features that have previously been related with motion: pulse clarity, and low and high-frequency spectral flux [3], [27], [28]. Pulse clarity estimates how clear the beat of the music is [30], and was computed using frame decomposition with the default values for window length (5 s) and hop length (500 ms). Spectral flux quantifies the amount that the frequency spectrum of a signal changes over time. The sub-band flux was computed by splitting it into 10 bands of 1 octave each, and using the frame decomposition method as in Alluri and Toiviainen comparing consecutive frames with a window length of 25 ms half overlapping (i.e. hop length of 12.5 ms) [30]. Low-frequency spectral flux has previously been related to greater speed of head movement [27] and more entrainment to the beat and bar levels of the musical rhythms [3]. In accordance with previous literature, low-frequency spectral flux was defined by the frequency ranges 50–100 Hz (sub-band no. 2; [27]) and 100–200 Hz (sub-band no. 3; [28]). We chose to examine both sub-bands of low-frequency spectral flux to ensure we captured any existing relations between these musical features and the motion. High-frequency spectral flux was defined as the range 6400–12800 Hz (sub-band no. 9) and has previously been related to greater speed of head and hand movement and amount of movement [27]. This sub-band flux represents rhythmic information in sounds resembling cymbals and hi-hats.

2.6 Analysis

QoM and musical features were compared using Spearman correlations. To compare these measures, both time series were resampled to 2 Hz. Time series were trimmed to remove the first 30-seconds. Spearman correlations were conducted for each point in the time series. Several participants had data loss events and rather than filling these motion gaps when resampling, any gaps in the original signal that were greater than 1 second were removed from the analysis.

3. RESULTS

3.1 Participants

Due to issues with the versions of the app at the time, there were several data loss events and challenges, especially with the Apple version of the application. In particular, Apple does not permit background recording, and therefore there are gaps in the data if the screen turns off or if the user navigates away from the MusicLab App. Certain Android phones also have battery optimization procedures that prevent background recording unless an application is given permission.

Figure 2: Geolocation data was collected from 14 participants

There were 7 participants with accelerometer data during Renick Bell's performance. Exclusion criteria included alignment issues, unrealistic human motion (complete stillness), and other data issues. One of the participants began recording after the start of the performance therefore a shake was not measured within an appropriate timeframe. Another participant had a shake that was much later than the other participants. Since this makes the

alignment questionable, we excluded them for this analysis. Therefore, the sample size of Renick Bell's audience is n = 5 for this analysis. Of these participants, n = 4 filled the pre-concert survey and the post-performance survey for Renick. Due to the small sample sizes, this paper should be interpreted as a proof of concept and no conclusions on the relation between motion and musical features can be drawn.

There were 7 participants with accelerometer data during Khoparzi's performance. One of the participants had a sampling frequency of 9 Hz on average, which is much lower than the other participants and it appeared as though the phone was static as if it wasn't secured to a body, therefore this participant was excluded. Another participant had three shakes in very close proximity. Since it was challenging to guarantee proper alignment, this participant was excluded. Another participant had only 4 minutes of data. Therefore, the final sample size of Khoparzi's audience is n = 4 for this analysis. Of these participants, all 4 filled the pre-concert survey and the post-performance survey for Khoparzi. Two participants had data during both Renick's and Khoparzi's performances.

3.2 Alignment Method Assessment

The alignment method could be improved. Due to the nature of the signal, the solution that worked best for identifying the shake was visual inspection. Automating alignment could improve the functionality of the MusicLab app, especially if there was a larger amount of data.

3.3 Music-related Motion

Correlations were conducted on each individual for illustrative purposes. Due to the small sample size, results should be interpreted with caution and should simply serve as a proof of concept. The correlations between quantity of motion and audio features are generally very low.

Renick	**PC**	**Low**	**Low 2**	**High**
1	-0.024	0.067	0.034	-0.019
2	-0.0082	0.013	-0.023	0.049
3	0.1	0.17	0.13	0.14
4	-0.031	-0.018	-0.019	-0.026
5	-0.048	-0.067	-0.055	0.016
Khoparzi	**PC**	**Low**	**Low 2**	**High**
2	0.18	0.042	0.092	0.11
4	0.098	0.024	0.038	0.13
6	0.07	0.048	0.066	-0.022
7	-0.13	-0.12	-0.11	0.097

Table 1: Correlations between quantity of motion and musical features

3.3.1 Pulse Clarity

In Renick's audience, one participant demonstrated a positive correlation between quantity of motion and pulse clarity. Interestingly, this participant reported that he enjoyed the performance more than all other participants (level of 6 out of 7; average level of enjoyment of other participants: 4), but he did not feel like he was moving to the beat (2/7).

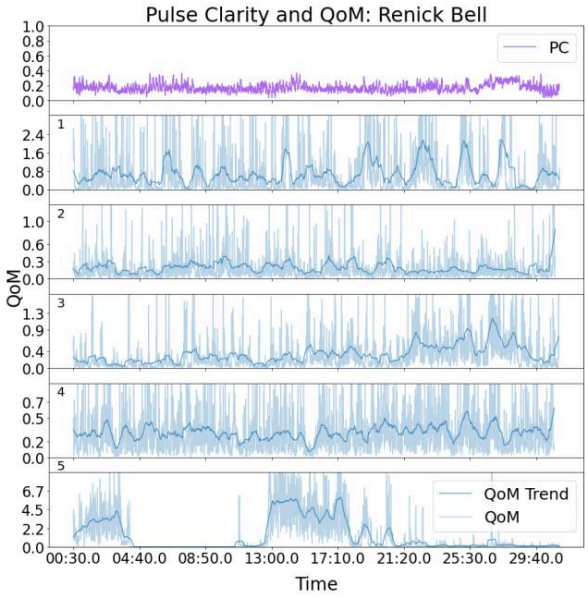


Figure 3: Pulse clarity and quantity of motion for each participant (1, 2, 3, 4, and 5), during Renick Bell's performance. Note that the y-axis is scaled to 1.5 x the maximum QoM trend for each participant and that some QoM peaks are cut off by the scaling. This scaling allows visualization of the variability within a single participant, but refer to the y-axis for the variability between participants.

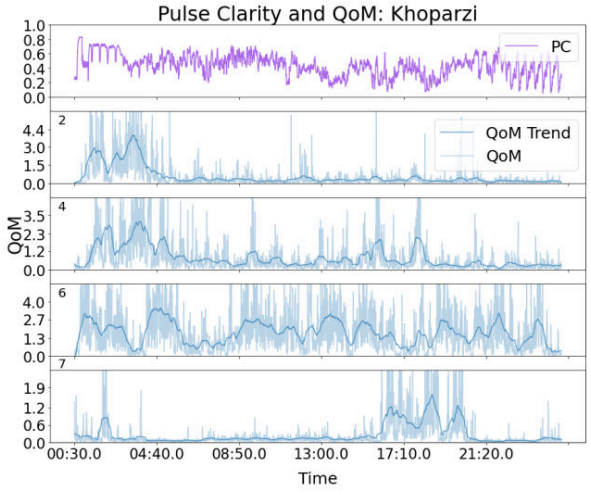


Figure 4: Pulse clarity and quantity of motion for each participant (2, 4, 6, and 7) during Khoparzi's performance. Note that the y-axis is scaled to 1.5 x the maximum QoM trend for each participant and that some QoM peaks are cut off by the scaling. This scaling allows visualization of the variability within a single participant, but refer to the y-axis for the variability between participants.

In Khoparzi's audience, three participants demonstrated a positive correlation between quantity of motion and pulse clarity, however one participant had a negative correlation. When visually inspecting this participant's motion, it appears that they were very still for most of the performance but moved around the middle of the performance. Unfortunately, there is no post-performance questionnaire for the participant therefore we are unable to investigate if this participant could have been sitting in the first half of the concert.

Spectral flux

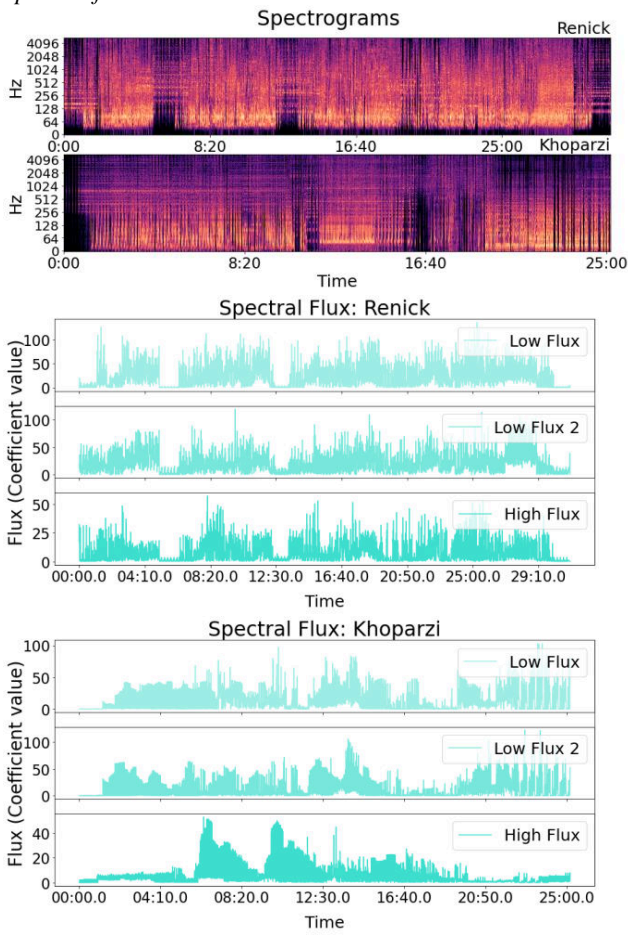


Figure 5: Spectrograms of each audio clip and spectral flux in low and high frequency flux bands for Renick and Khoparzi's performances. Note that for spectral flux, the y-axis is scaled to each sub-band, so refer to the y-axes for the variability between performers.

In Renick's audience, two participants demonstrated a positive correlation between QoM and low frequency spectral flux. The participant that showed the positive correlation with pulse clarity is the same with the strongest correlation in the low frequency sub-band. In the higher low frequency sub-band (band no. 3), this same participant has the strongest correlation again. For high frequency spectral flux, there were two participants with positive correlations, the same participant as before and another participant that did not show correlations in the low frequency flux.

In Khoparzi's audience, one audience member showed a negative correlation between QoM and low frequency flux (band no. 2). In the higher low frequency flux, two participants demonstrated positive correlations and the same participant as before demonstrated a negative correlation.

Khoparzi had a higher level of pulse clarity than Renick. The differences in pulse clarity are also easily heard in the audio, where there are more irregular rhythms during Renick's performance. Renick performed with more spectral flux in all frequency bands. In addition to the performances being quantitatively different, they were qualitatively different featuring different timbres rhythmic components, and coding/musical styles.

3.4 Shared Location

Through data exploration, it was observed that two individuals (participants 2 and 4) had the same geolocation and they indeed reported that they were watching the performance together with another person. They both had motion data during Khoparzi's performance therefore we calculated a Pearson correlation between their quantity of motion (r = .26). The other participants' quantity of motion were much less correlated (r ranging from -.08 to .01). A visual inspection of the motion indicated that it may be that one of their motion datasets was misaligned because shifting one later by 17 seconds aligned the signals even better for a Pearson correlation of r = .43.

Figure 5: Two participants watched the concert together and showed high correlation between their quantity of motion. The trends are displayed here normalized for each participant. However, the correlation was improved by shifting the data in time which may indicate that the alignment method could be improved in future research.

It appears that the participant indicated in blue may have begun recording too late and thus shook their phone too late such that the first sound actually happened before the participant began recording. This further indicates that the alignment method used for this experiment was flawed and taking care to ensure that participants are warned well

in advance of when to begin their motion recording should prevent situations like this.

4. DISCUSSION

The MusicLab Lockdown Rave concert experiment served as a proof of concept showing that remote motion measurement is indeed a useful way of measuring music-induced motion. Accelerometer data collected from the inertial measurement units of participants' own smartphones can be converted into quantity of motion data and subsequently compared to musical features, or to other individuals' movement.

The analysis presented here indicates two important components of working with these types of applications. The alignment method conducted for this experiment proved to be challenging to automate during analysis. It is also prone to participant error since it relied on participants reacting quickly to a sound. One solution would be to simultaneously record audio from the participants' own smartphones. While an audio synchronization might be convenient, privacy concerns, technical complications, and app distributor constraints render this an untenable strategy at this time.

Through trying to solve this alignment challenge, we developed a new strategy in which participants tap on their smartphones to a recording of an isochronous beat at two distinct tempi. Our testing shows that the tap of a finger on the phone produces sufficient acceleration to be detected in the signal. However, due to participant errors when tapping, such a method still requires some visual inspection to ensure that alignment occurs properly.

Previous research indicated that movement can be influenced by musical features including pulse clarity and high and low spectral flux, with more flux and pulse clarity leading to greater quantity of motion [3].The differences between the results presented here and the existing literature could be attributed to several factors including differences in the variability of pulse clarity in the stimuli. The live-coded algorave music that was improvised at this concert was a different genre than previously explored. Each performer used a distinct system for creating their music and the differences in sonic texture and rhythmicity between them and from music that has typically been examined in relation to music-induced motion may explain the observed differences.

In previous research participants' motion was measured in a laboratory context which offers more experimental control than measuring motion in participants' homes. Despite experimenter instructions and participants' best intentions, participants may be distracted from their task due to various interruptions that occur regularly in home life.

Virtual concerts are also very different experiences than live concerts due to differences in social and environmental factors. Due to social distancing guidelines, people may be watching virtual concerts alone rather than in groups and this would create a different propensity for movement. Research suggests that perceiving movement activates the same regions of the brain that are involved in producing the movement [31]. Therefore, if we are in the presence of others and perceiving others' actions, we may be more likely to move as well.

The social element of music listening, even during virtual concerts was shown by comparing two individuals who shared the concert experience together. The similarity between their quantity of motion may be indicative of similar movements. Given that motor entrainment promotes prosocial benefits [32], even when the entrainment occurs in virtual reality [33], it is possible that sharing virtual concert experiences could promote bonding, especially during times of social distancing. Indeed research supports that viewing livestreamed, but not pre-recorded virtual concerts promotes social connectedness and alleviates feelings of loneliness [34], [35]. Whether movement plays a role in feelings of togetherness at a virtual concert remains unexplored, however the MusicLab App is a useful tool for examining movement remotely.

After the MusicLab Algorave project, development on the application and the protocols associated with data collection continued based on the issues that arose. Rather than a pre- and post-concert survey, now up to 13 surveys can be administered at the same event. To reduce data loss during events, we have implemented a function so that data is submitted as 1-minute packages. In new app versions, surveys and motion data are also linked more reliably. Participants can withdraw their participation directly in the App. There are still some limitations including not being able to record motion in the background on all Apple and some Android devices. However, we implemented a screen dimming functionality that reduces screen brightness when data is being recorded. This feature will not only help prolong battery life, but it will also help make screens less disturbing to concertgoers.

Future applications of the MusicLab App include usage at live, hybrid, and virtual musical events, examination of coordination between audience members, and further exploration in different musical and ecologically valid contexts. The motion could also undergo more elaborate analyses than quantity of motion including periodicity or frequency-based analyses.

As a result of the improvements that were implemented through challenges discovered during the MusicLab Algorave project, the MusicLab App was ready for the next project. MusicLab Copenhagen was a large concert in which the Danish String Quartet performed to live and livestreaming audiences. There we successfully captured data from 79 participants in the live audience and 34 participants in the virtual audience. Adjustable phone holders were purchased so that the phones could sit high on wearers' chests, which provided a uniform position for recording across the live audience. Surveys were provided in the app in both English and Danish for livestreaming viewers to choose their preferred language. The MusicLab App has already proved itself a formidable tool for measuring motion remotely at virtual concerts and together in a live concert hall.

5. CONCLUSIONS

The MusicLab App is a mobile application developed to measure motion and collect surveys in musical audiences, who may be viewing a live concert together, or a virtual

concert apart. It leverages the inertial measurement unit (IMU) sensor to track acceleration, rotation, and geolocation. Audience motion and survey responses can effectively be collected and used to understand how participants engage with and experience virtual concerts. The MusicLab App has undergone a number of improvements and has proved itself as a viable tool in the embodied music cognition researcher's toolkit.

Acknowledgments

Pedro Pablo Lucas Bravo is the developer who implemented the most recent changes to the MusicLab App. Augusto Dias was the original developer and prepared the app for MusicLab Algorave. Solveig Sørbø helped with organization and hosted the event. Kayla Burnim assisted with moderating the YouTube chat comments and supported data management, data processing approvals, and further app feature requests. Qichao Lan spoke as an expert on live-coding during the panel discussion. Rebecca Josefine Five Bergstrøm provided legal support from the University Library.

This work was supported by the Research Council of Norway through its Centers of Excellence scheme, project number 262762, NordForsk's Nordic University Hub, project number 86892, and Dana Swarbrick is supported in part by funding from the Social Sciences and Humanities Research Council Doctoral Fellowship.

6. REFERENCES

[1] V. E. Gonzalez-sanchez, A. Zelechowska, and A. R. Jensenius, "Correspondences Between Music and Involuntary Human Micromotion During Standstill," *Front. Psychol.*, vol. 9, no. 1382, pp. 1–10, 2018.

[2] R. T. Solberg and A. R. Jensenius, "Group behaviour and interpersonal synchronization to electronic dance music," *Music. Sci.*, vol. 23, no. 1, pp. 111–134, 2019.

[3] B. Burger, M. R. Thompson, G. Luck, S. H. Saarikallio, and P. Toiviainen, "Hunting for the beat in the body: on period and phase locking in music-induced movement," *Front. Hum. Neurosci.*, vol. 8, no. November, pp. 1–16, 2014.

[4] B. Burger, S. H. Saarikallio, G. Luck, M. R. Thompson, and P. Toiviainen, "Emotions Move Us : Basic Emotions in Music Influence People ' s Movement to Music," *Proc. 12th Int. Conf. Music Percept. Cogn. 8th Trienn. Conf. Eur. Soc. Cogn. Sci. Music*, no. November 2014, pp. 177–182, 2012.

[5] A. Zelechowska, V. E. G. Sanchez, B. Laeng, J. K. Vuoskoski, and A. R. Jensenius, "Who Moves to Music ? Empathic Concern Predicts Spontaneous Movement Responses to Rhythm and Music," vol. 3, pp. 1–16, 2020.

[6] E. Carlson, B. Burger, J. London, M. R. Thompson, and P. Toiviainen, "Conscientiousness and Extraversion relate to responsiveness to tempo in dance," *Hum. Mov. Sci.*, vol. 49, pp. 315–325, 2016.

[7] E. Carlson, B. Burger, and P. Toiviainen, "Dance Like Someone is Watching: A Social Relations Model Study of Music-Induced Movement," *Music Sci.*, vol. 1, p. 205920431880784, 2018.

[8] D. Swarbrick *et al.*, "How live music moves us: Head movement differences in audiences to live versus recorded music," *Front. Psychol.*, vol. 9, no. 2682, pp. 1–11, 2019.

[9] P. Janata, S. T. Tomic, and J. M. Haberman, "Sensorimotor Coupling in Music and the Psychology of the Groove," *J. Exp. Psychol. Gen.*, vol. 141, no. 1, pp. 54–75, 2011.

[10] P. J. Maes, M. Leman, C. Palmer, and M. M. Wanderley, "Action-based effects on music perception," *Front. Psychol.*, vol. 4, no. JAN, pp. 1–14, 2014.

[11] J. Phillips-silver and L. J. Trainor, "Hearing what the body feels : Auditory encoding of rhythmic movement," vol. 105, pp. 533–546, 2007.

[12] F. Manning and M. Schutz, "' Moving to the beat ' improves timing perception," pp. 1133–1139, 2013.

[13] S. Nozaradan, I. Peretz, M. Missal, and A. Mouraux, "Tagging the Neuronal Entrainment to Beat and Meter," *J. Neurosci.*, vol. 31, no. 28, pp. 10234–10240, 2011.

[14] M. Leman, P.-J. Maes, L. Nijs, and E. Van Dyck, "Embodied Music Cognition," in *Springer Handbook of Systematic Musicology*, R. Bader, Ed. Springer, 2018.

[15] M. Leman, *Embodied music cognition and mediation technology*. MIT Press, 2008.

[16] I. Cross, "Music and social being," *Musicol. Aust.*, vol. 28, no. 1, pp. 114–126, 2005.

[17] N. Moran, "Social implications arise in embodied music cognition research which can counter musicological 'individualism,'" *Front. Psychol.*, vol. 5, no. JUL, pp. 1–10, 2014.

[18] A. R. Jensenius, "Methods for Studying Music-Related Body Motion," *Springer Handbooks*, pp. 805–818, 2018.

[19] K. H. Hampton *et al.*, "Mapping health data: Improved privacy protection with donut method geomasking," *Am. J. Epidemiol.*, vol. 172, no. 9, pp. 1062–1069, 2010.

[20] R. T. Solberg and A. R. Jensenius, "Optical or inertial? Evaluation of two motion capture systems for studies of dancing to electronic dance music," in *SMC 2016*, 2016.

[21] A. R. Jensenius, "Best versus Good Enough Practices for Open Music Research," *Empir.*

Musicol. Rev., vol. 16, no. 1, pp. 5–15, 2021.

[22] E. Mas-Herrero, J. Marco-Pallares, U. Lorenzo-Seva, R. J. Zatorre, and A. Rodriguez-Fornells, "Individual Differences in Music Reward Experiences," *Music Percept.*, vol. 31, no. 2, pp. 118–138, Dec. 2013.

[23] S. D. Gosling, P. J. Rentfrow, and W. B. Swann, "A very brief measure of the Big-Five personality domains," *J. Res. Pers.*, vol. 37, no. 6, pp. 504–528, 2003.

[24] M. H. Davis, "A Mulitdimensional Approach to Individual Differences in Empathy," *Cat. Sel. Doc. Psychol.*, vol. 10, no. MS. 2124, p. 85, 1980.

[25] T. Kelkar and A. R. Jensenius, "Analyzing free-hand sound-tracings of melodic phrases," *Appl. Sci.*, vol. 8, no. 1, 2018.

[26] O. Lartillot and P. Toiviainen, "A MATLAB toolbox for musical feature extraction from audio," in *Proceedings of the 10th International Conference on Music Information Retrieval, ISMIR 2007*, 2007, pp. 127–130.

[27] B. Burger, M. R. Thompson, G. Luck, S. Saarikallio, and P. Toiviainen, "Influences of Rhythm- and Timbre-Related Musical Features on Characteristics of Music-Induced Movement," *Front. Psychol.*, vol. 4, no. 183, 2013.

[28] B. Burger, J. London, M. R. Thompson, and P. Toiviainen, "Synchronization to metrical levels in music depends on low-frequency spectral components and tempo," *Psychol. Res.*, vol. 82, no. 6, pp. 1195–1211, 2018.

[29] O. Lartillot, T. Eerola, P. Toiviainen, and J. Fornari, "Multi-feature modeling of pulse clarity: Design, validation and optimization," *ISMIR 2008 - 9th Int. Conf. Music Inf. Retr.*, pp. 521–526, 2008.

[30] V. Alluri and P. Toiviainen, "Exploring perceptual and acoustical correlates of poly-phonic timbre," *Music Percept.*, vol. 27, no. 3, pp. 223–242, 2010.

[31] V. Gallese, L. Fadiga, L. Fogassi, and G. Rizzolatti, "Action recognition in the premotor cortex.," *Brain*, vol. 119 (Pt 2), pp. 593–609, Apr. 1996.

[32] L. Cross, M. Turgeon, and G. Atherton, "How Moving Together Binds Us Together: The Social Consequences of Interpersonal Entrainment and Group Processes," *Open Psychol.*, vol. 1, no. 1, pp. 273–302, 2019.

[33] B. Tarr, M. Slater, and E. Cohen, "Synchrony and social connection in immersive Virtual Reality," *Sci. Rep.*, vol. 8, no. 1, pp. 1–8, 2018.

[34] D. Swarbrick, B. Seibt, N. Grinspun, and J. K. Vuoskoski, "Corona Concerts: The Effect of Virtual Concert Characteristics on Social Connection and Kama Muta," *Front. Psychol.*, vol. 12, no. June, pp. 1–21, 2021.

[35] K. E. Onderdijk *et al.*, "Livestream Experiments: The Role of COVID-19, Agency, Presence, and Social Context in Facilitating Social Connectedness," *Front. Psychol.*, vol. 12, no. May, pp. 1–25, 2021.

DESIGNING THE (RE)CONTEXT: STOLEN GOODS (STOCKETUS)

Johannes (Jos) Mulder
School of Music, The Australian National University
Jos.mulder@anu.edu.au

ABSTRACT

This paper discusses my work, *Stolen goods (Stocketus)* from 2017 for ensemble, digital electronics and 'pile of loudspeakers'.[1] The work was designed to juxtapose acoustic musical instruments and the loudspeakers (and microphones) that are so often used to project performances to the audience. By structuring the composition around the insertion of a descending time delay (going down from five to zero seconds over the duration of the work), a performance of the work explores different sonic interactions between the ensemble and the loudspeakers, for instance, creating echoes, rhythmical patterns, or simple phasing effects. In my research I am interested in how the use of technology operationalizes the performance context as a musical element rather than as a frame for a concert. Amplification technology can recontextualize musical performances and the use of music technology for this work was designed with this recontextualization in mind. The paper will address both aesthetic as well as practical considerations, e.g. notation and synchronization of the score, electronics and performers, and finishes with plans for future works.

1. INTRODUCTION

Earlier in my career I had many opportunities to work as a music technologist, specifically in the field of live sound reinforcement.[2] I mainly focused on performances of 20th century avantgarde electroacoustic works, that are usually supported by amplification systems either to enhance acoustic balances or to project (live) electronic sound sources. I have since segued into an academic career focusing my research on this specific topic. I remain fascinated by the combination of instrumental and loudspeaker sources and the challenges and wonders it provides within acoustic spaces but also in our perception, as often the sound we hear does not emanate from the instrument we see but from a loudspeaker on a corner of the stage. At the same time amplification systems play a large role in contextualizing music for instance by allowing music of small ensemble in large venues for large audiences, or for symphony orchestras to perform outside of the specific acoustic conditions of concert halls (e.g. on an outdoor stage).

During my doctoral studies (until 2013) I considered many different approaches of designing experiments that take these elements of perception and contextualization into account. But the first real opportunity to do this came in 2017 when ensemble *decibel new music* commissioned and supported me to create and perform a work exploring this topic. The research question in this iteration was very much a *how* question: how can I design a system to musically explore the interactions set out above? Moving forward from this question, in the future, different research questions may be explored.

As such the work described in this paper is not the outcome of a somewhat linear composition process but the first in a few iterations around an experimental system of performers, microphones, delay lines, amplifiers, and loudspeakers as well as, of equal importance, a score that allows all these parts to synchronize.

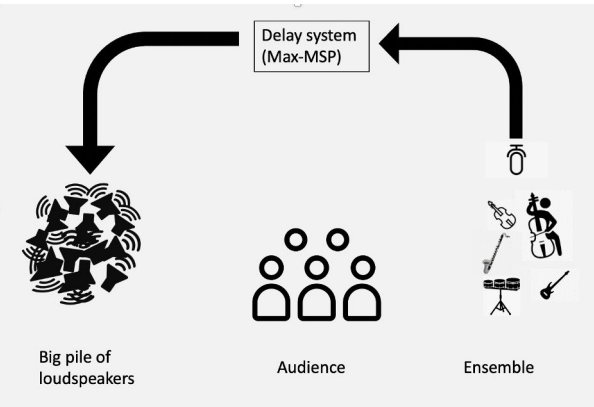


Figure 1. Initial design sketch for the performance system.

Copyright: © 2022 Johannes Mulder. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any médium, provided the original author and source are credited.

[1] A brief introduction to and recording of the work are available on Australian public radio (ABC) New Waves Podcast: https://www.abc.net.au/classic/programs/new-waves/decibel-electronic-concertos-act-1/9361976

[2] Because of the autobiographical nature of the research described in this paper, it is written in the first person.

2. STOCKETUS

Stolen Goods *Stocketus* was a commission by decibel new music, the Australian new music ensemble at the time based in Perth, led by Professor Cat Hope.[3] The work created for six musicians (five acoustic instruments and one electronic performer) and a 'pile of loudspeakers'. The work's electroacoustic concept was shaped around the delay process, with a five second delay that would gradually reduce to zero over the duration of the work.

Initially the concept for the work was an installation/designed soundscape, with a big pile of loudspeakers in the middle of a gallery space and performers and audience o opposing sides (fig. 1). When the work was ultimately performed it took place on a traditional stage and for that occasion, we set-up a number (a 'pile') of loudspeakers center stage with the performers in a U shape around it (see fig. 3). Ensemble decibel new music's scoreplayer[4] (an iPad-based scrolling score player and creation environment [1, 2]) was used as a score as well as a way to sync to the changing delay times which were realized using MAX/MSP. Microphones transduced the sounds from the instruments (flute, clarinet, viola, violincello, percussion) which were mixed into a single signal that was fed through the delay process and into the pile of stacked loudspeakers, with the sixth performer realizing the electronics. The pile of speakers was given its own voice by emanating a soundscape that was created applying granular synthesis using, unrecognizable, recordings of rehearsals of the work. This soundscape was playing as a prelude while the audience entered the room to metaphorically, give the loudspeakers their own voice. It played throughout the piece loosely improvised by the electronic performer, in response to graphical cues in the score.

2.1 Concept

The concept behind this setup served two purposes, firstly, liberating the loudspeakers, so crucial for so many canonical works, from the dark corners of the stage, and so moving away from the typical sound staging. Secondly, the shifting delay times, over the course of the work, change the perceived relation between the instrumental sound and the loudspeaker sound. A five second delay, is rather long and relating the initial and the delayed sound to each other in a rhythmical way is challenging and somewhat estranging. The composition explores this space with clear incidental sounds, for instance a stroke of single bell, which is echoed in the loudspeakers five seconds later, doubled by the clarinet, and again by the strings instruments another delay period later. By and by the interaction becomes more musical and rhythmical, not unlike a slow *hoquet*. That particular element returns in the work's name: *Stoketus* is a combination of Louis Andriessen's seminal work *Hoketus* (1976) and *Stockie* an obscure nickname for Karlheinz Stockhausen, who is well-known for original work using tape delays, particularly in Solo (1967) [3]. *Stolen goods* refers to the nature of delay systems (notably echoes), stealing a sound and releasing it a while later, but also the nature of music creation, borrowing (or standing in shoulders) of earlier music makers. This notion popped up in many facets of the work, for instance in its duration which was 273 seconds (more popularly known as four minutes and thirty-three seconds).

3. BACKGROUND

The role of loudspeaker amplification in avantgarde music has been of interest to several authors [4-6]. Pertinent here is Simon Emmerson's [7] list of functions of amplification (table 1). Elsewhere [8] I have argued that such functions can be understood along a continuum of relative amplification levels. Subtle balancing requires lower amplification levels in comparison to intentional loudspeaker distortion or generating feedback.

Function	Example
Balance	Fixing Acoustic imbalances
Blend	Integrating timbres
Projection	Zooming in/Intentional dislocation
Perspective	Zooming out/ dislocation of space and time
Coloration	Intentional Distortion
Resonance	Intentional Feedback

Table 1. Simon Emmerson's six functions of loudspeaker amplification

Similarly, a continuum of relative runtime differences between acoustic source and amplified (loudspeaker) source can be considered. On a level almost outside of our perception there is an interesting phenomenon occurs: sound in its transduced form as an analogue electronic signal travels roughly a million times faster than sound in air. Consequentially, even though our perception is very well able to fuse what we hear with what we see, including for instance acoustic reflections arriving at different times, the misalignment of time is a given in electro acoustic music.

In sound system design and engineering small amounts of time delay can be introduced to realize subtle time alignment between different loudspeaker systems [9]. Time delays in sound system design can also be applied to delay the loudspeaker sound somewhat with respect to acoustic sources on stage, making use of the precedence (or Haas) effect to support localization of on-stage sources, from an audience perspective.

In an ideal world I would aspire to replicate, in the context of a concert, an elegant experiment described by Lewald and Guski [10]. To learn more about auditory visual integration they used led lights and sound sources at different distances away (between 1 and 50 meters), in an outdoor setting. Participants were asked to press one of two keys, depending on whether the sound or the light was perceived first. The outcomes presented in that publication suggest that there is an 'Horizon of Simultaneity' where the sound and light stimuli are perceived synchronously up to around a distance of 20 meters, however the nature of crossmodal perception (i.e. stimuli appear as one unified event rather than two different percepts) may extend this horizon to greater distances, and perhaps as much as 100

[3] https://decibelnewmusic.com/

[4] https://decibelnewmusic.com/decibel-scoreplayer/

milliseconds in temporal disparity. The work discussed in this paper is a first attempt to explore this idea of auditory visual integration in the complexity of acoustic and loudspeaker sources and in the context of (live) musical performance.

4. CREATING THE SCORE

4.1 Preparation

Firstly, I should mention that even though I have worked as a music technologist and musician for three decades this is the first formal composition I have created and the project both challenged my existing skill sets as well as supported the development of new skills.

The scrolling score environment provides a great framework to create a work like this. I used a mix of traditional notation and graphics, leaving room for the performers to choose pitches and timbres while the timing was stricter to work with the delay system. When I began drafting the score I used a DAW, to also allow for the creation of a click track, should this be necessary. For the greatest part of the work, I opted to use traditional rhythmic notation with varying meters and tempo to make the inevitable acceleration caused by the delay system more gradual. However, this was at odds with the systematic linear reduction in delay time, which I consequentially made more gradually, reducing the delay time in a series of steps, of varying size. This change resulted in the creation process becoming more of a puzzle for which the pieces needed a design approach to come together. For most parts of the work, I wanted the source and delayed notes to be working in rhythmical time, which in theory, would allow me to use compositorial devices such as canons and fugues.

An interesting side effect of this rather rational and linear approach to meters and tempi in the score is that it allowed me, perhaps even encouraged me, to be very liberal with assigning rhythmical and pitch values.

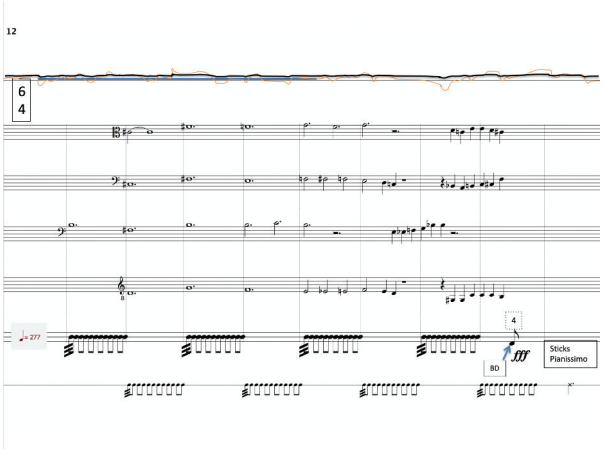


Figure 2. Page 12 of the scrolling score, tempo =277, delay 1302 milliseconds. Top to bottom: Electronic part (granular soundscape), Vla., Vc., Bass, Bcl., Perc., delay response.

For instance (figure 2) a section with half and whole notes played on the instruments are sounding in canon through the delay system, the duration of one bar later. The lowest system shows the echoes of the drum roles played on the one but last system. The pitch values and rhythms where simply improvised on a keyboard and recorded as midi data.

4.2 A Rational Path to Chaos

Using an increasingly complex spreadsheet (table 2) I optimized the size of the descending steps to be maximally useful using common (as in, somewhat executable) meters. With a combination of guesswork and mathematics I derived the delay times for each part with one part on one page. Each of 21 pages, conveniently 273 is divisible by 13, had the same duration of 13 seconds, to realize a consistent scroll. The delay times where matched to BPM and meter in a way that would allow for a reasonable number of either bars or notes per page. A BPM greater than 300 would be very problematic to play and for the final two pages I had to switch to a different notation strategy.

4.3 Fast-paced Notation

Initially I borrowed from Jazz, using rhythmical Big Band figures or patterns. To increase the information density on each page every individual quarter note in the score was to be performed as a predefined 16th or triplet figure of four (or three) notes. In rehearsals this proved to be cumbersome and after that I 'stole' once more from Louis Andriessen who developed (e.g. in *Workers Union* from 1975) the notion of 'pitch contours', basically free pitch, but the melodic shape (descending or ascending) is scored with the notated rhythm. With inevitable cacophony, if somewhat planned, in the final three pages of the score, I swapped from stepping down the delay time per page to a gradual descent from 100 milliseconds to zero. This brought back the initial idea of sliding through a delay continuum, starting at 100 milliseconds (indicative of the auditory-visual integration window mentioned by Lewald and Guski above [10]) going down through phasing effects and ending with no delay at all.

Page	Delay time	BPM Delay	Meter	BPM Score	Beats in 13 sec.	Bars per page	Remarks
0	5000	12	9	108	23.4	2.6	
1	4330	13.9	9	125	27.1	3.01	
2	3900	15.4	8	123	26.7	5.01	Dotted
3	3700	16.2	6	97	21	3.5	
4	3260	18.4	9	166	36	4	
5	3007	20	6	120	26	4.33	
6	2790	21.5	12	258	55.9	4.66	
7	2600	23.1	11	254	55	5	
8	2440	24.6	8	197	42.7	8.01	Dotted
9	2168	27.7	10	277	60	6	
10	1860	32.3	7	226	49	7	
11	1500	40	4	160	34.7	13.01	Dotted
12	1302	46.1	6	277	60	10	
13	1000	60	3	180	39	13	
14	766	78.3	2	157	34	17	
15	500	120	1	120	26	26	
16	250	240	1	240	52	52	
17	200	300	1	300	65	65	
18	100	600	1	600	130	130	
19	50	1200	1	1200	260	260	
20	Chaos						

Table 2. Spreadsheet to analyze delay times, BPM and meter for each page in the score.

4.4 Scrolling Score

Transferring the score from the notation module in the DAW (I used Cubase) to the scoreplayer was a challenge

and a considerable amount of editing was needed to fit the individual pages I created (one page for each step down in delay time) in one long *scroll*. After a first rehearsal, with the ensemble (who, needless to say, are very experienced with the scrolling score system), it became obvious that the combination of traditional rhythmic notation and the scrolling score did not allow for precise execution at higher tempos. But at this stage in the project there was no time to start from scratch. In rehearsal we did experiment with a click track, which would perhaps allow for a smoother execution, but would also require more rehearsal time. Additionally, playing with a click track on headphones can be unpleasant for performers and can also be audible to the audience in quieter parts.

When considering how this scrolling score shapes the performance ecology, synchronization is a key element. Using OSC, the score communicates with the MAX/MSP patch that realized the audio delay, and so synchronizing not just the performers but also the audio processing [11].

5. PERFORMANCES

The work was premiered at the Totally Huge New Music Festival in Perth, Western Australia in October 2017. The venue at the Perth Institute of Contemporary Arts has the layout of a traditional theater, and the set-up was adapted from the initial sketch (fig. 2), with the 'pile of loudspeakers' center stage, now more a stack than a pile (fig. 3).

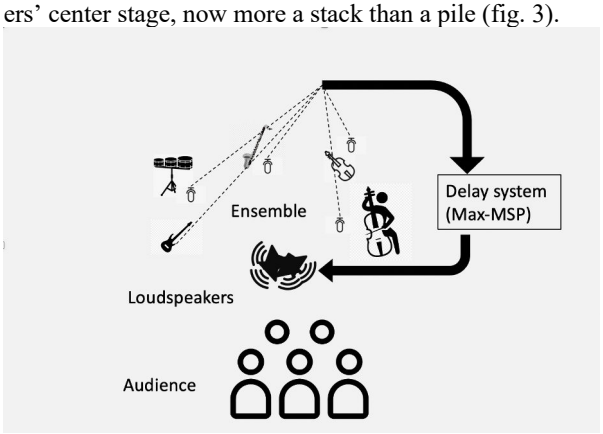


Figure 3. Performance set-up at PICA, Perth 2017.

Figure 4. Concert at PICA, left to right Cat Hope, Lindsay Vickery, Aaron Wyatt, Tristen Parr, loudspeakers and Stuart James. Out of shot, on the left is percussionist Louise Devenish. Photo: Bohdan Warchomij.

The centrality of, and the spotlights on the loudspeaker steak also became an interesting variation on the approach that ensemble decibel usually takes, with a loudspeaker near each performer (fig. 4).

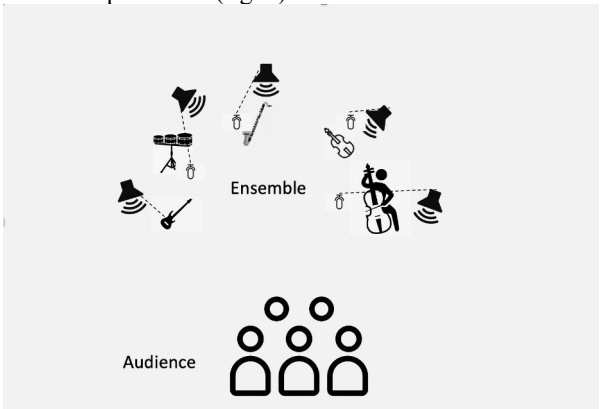


Figure 5. Stage layout and loudspeaker set-up generally used by ensemble decibel.

During this first performance the WIFI router connecting the individual iPads in use by the performers crashed, or otherwise disconnected, which instantly turned the performance into an improvised version of the work. It left the audience, and the composer, somewhat confused about what they experienced, and the response was unsurprisingly lackluster. Consequentially, the recording of the performance was useless as a document, but luckily the ensemble agreed to recording a performance of the work in the studio. This new context for the work also allowed for a bypass of the tempo issue described above, with the final two pages of the score now printed out, making the execution easier and very much more precise.

In the podcast (see footnote 1) documenting both the concert and the studio recording, producer and presenter Stephen Adams describes the studio recording of the work as follows: "…with everything in sync you [got to] hear the strange temporal shifts in the relationship between the live players and the speaker stack […] At first sounding far apart in time like two unconnected ensembles, then, in tighter, out of phase rhythms as the speaker delay shortened. And finally, the ensemble and speakers lining up with each other to produce what we casually call amplification. […] a reminder that amplification is anything but an ordinary acoustic situation, the speakers in the room are effectively a whole other instrument, an electronic performer."

The work was performed once more at the 2018 Australasian Computer Music Conference at the Western Australian Academy of Performance arts in Perth, again using a paper score for the problematic final pages. This performance was much better received and the intended effects were much easier to note, and commented upon by several audience members.

5.1 Outcomes

In answer to the research question raised above (how can I design a system to musically explore the contextualization realized by electroacoustic amplification?), the challenges to perform faster tempi at low delay times using the score player is a first outcome of this project. It is certainly of interest that, although perhaps this should not come as a surprise, that once the delay time is below 100 milliseconds (i.e. the final pages of the score) the execution of the score becomes a problem, and a different approach to the composition is required.

Another way of considering this challenge in the current approach is that the pages of the score scroll at the same speed, whereas a varying speed would allow for a reduction of the information density.

One way I have begun exploring is the using the so-called canvas mode in the score player application that allows the scrolling score to be used as a drawing surface using OSC commands.[5] This would allow for experimentation with response time to instant cues, potentially allowing for more precision at greater tempos, and or, varying the scrolling speed.

6. FUTURE WORK

As mentioned above, my intention with this work is the realization of an experimental system that allows for more formal, decontextualized, exploration of the relation between instrument and loudspeaker. To realize that I will need to develop several iterations of this work's score. As indicated, improvements in the score need to be explored, but also different strategies to realizing the rhythmical synchronization between the interpreted score and the delayed sounds. In the version discussed in this paper tempo, meter and rhythmical possibilities were decided by choices made around the delay system. A next version could do the opposite where an accelerating composition, for instance using rhythmic modulation and complex meters, informs the changing delay times.

Currently a version for Disklavier is in preparation to be used in the more formal setting of a recording studio. This will create an environment for experimental exploration of the subtle changes in balance (between acoustic sound from the piano and amplified sound from the loudspeaker) in relation to subtle changes in delay time of the transduced sound. This could be a version of the work exploring specifically the domain between zero and 100 milliseconds of time delay. Using a Disklavier would fix the performance (and scoring) parameters while other conditions (visibility, relative level etc.) can be varied. Participants can be tasked with balancing the level of the transduced sound coming from the loudspeaker and the acoustic sound as produce by the piano. The aim of such experimentation is an increase of our knowledge on the practice of music creation, the subtilities of auditory visual integration, and, for instance, the different functions of sound amplification as set out by Emmerson.

Acknowledgments

I would like to thank Louise Devenish, Aaron Wyatt, Tristen Parr and Lindsay Vickery of ensemble decibel new music for their energy in performing the work, as well as TURA new music. A special mention and thanks to Stuart Graham for fixing everything in the mix. And finally, to Cat Hope, director of the ensemble, composer and academic and inspirer of musos on the entire Australian continent and far beyond. Many thanks also to the reviewers and organizers of the conference.

7. REFERENCES

[1] Wyatt, A., L. Vickery, and S. James. "Unlocking the Decibel ScorePlayer", in *Proceedings of the International Conference on Technologies for Music Notation and Representation -- TENOR'19*. 2019. Monash University.

[2] Wyatt, A., et al. "Animated Music Notation on the iPad", in *Proceedings of ICMC*. 2013.

[3] Cancino, J. P. and J. Mulder, "On Stockhausen's Solo(s): Beyond Interpretation." *Leonardo Music Journal* 28:13-18. https:// doi: 10.1162/lmj_a_01036.

[4] Mulder, J. "The Loudspeaker as musical instrument", in *Proceedings of NIME* 2010, Sydney. http://doi.org/10.5281/zenodo.1177861

[5] Mulder, J., *Making things louder: amplified music and multimodality*, in *FASS*2013, University of Technology Sydney. http://hdl.handle.net/10453/21903.

[6] Van Eck, C., "Between air and electricity: microphones and loudspeakers as musical instruments". 2017: Bloomsbury Publishing USA.

[7] Emmerson, S., *Living electronic music*, 2007, Aldershot: Ashgate.

[8] Mulder, J. "Sound Amplification Technology and the Acousmatic Experience", in *Proceedings of Systematic Musicology*, 2009, Ghent.

[9] McCarthy, B., *Sound systems : design and optimization : modern techniques and tools for sound system design and alignment*, 1st ed. 2007, Oxford ; Burlington, MA: Focal.

[10] Lewald, J. and R. Guski, "Auditory-visual temporal integration as a function of distance: no compensation for sound-transmission time in human perception". *Neuroscience Letters*, 2004. 357(2): p. 119-122. https://doi.org/10.1016/j.neulet.2003.12.045

[5] https://www.psi-borg.org/#canvas

[11] Masu, R., N. Do Nascimento Correia, and T. Romao, "NIME scores: a systematic review of how scores have shaped performance ecologies in NIME", 2021. http://doi.org/10.5281/zenodo.1177861

TARTYP GENERATIVE GRAMMARS AND THE ANALYSIS OF HIP HOP BEATS

Israel Neuman
Texas Southern University
Israel.Neuman@tsu.edu

ABSTRACT

Pierre Schaeffer's *musique concrete* is considered to be the predecessor of digital sampling. The latter is a defining element in the development of hip hop. Drawing on historic and stylistic connections between these genres, we suggest the use of the Schaefferian core typology, the TARTYP, in the analysis of hip hop beats. We present a TARTYP classification of the beat's basic components, based on samples taken from the pre-mastered multitrack sessions of professionally produced hip hop songs, mapping these components to TARTYP sound object types. We use this classification to define sets of rewrite rules in the TARTYP generative grammars that we have presented in previous studies. We demonstrate how a path extracted from a rewrite rule set within the TARTYP Balanced grammar regenerates a representative hip hop drum pattern.

1. INTRODUCTION

The connection between hip hop music and Pierre Schaeffer's *musique concrete* is usually drawn from a technical perspective. *Musique concrete* is considered to be the predecessor of digital sampling. The latter is a defining element in the stylistic development of hip hop. In a historical review of sampling techniques Howell maintains that the precursors of the digital sampling method introduced by the Fairlight CMI in the 1970s were the sampling techniques used by Pierre Schaeffer in his *musique concrete* [1]. A similar idea was expressed by Tully in [2]:

> Sampling is the digital equivalent of music concrete, wherein common sounds are manipulated (and sometimes integrated with traditional instruments) to produce musical compositions.

While Tully and Howell are concerned with digital sampling as a technology, Schloss, in [3], sees it as a hip hop aesthetic, an underlying principle of this musical genre. However, is the connection between these two genres based only on technological developments or are there any aesthetical connections that can be made between *musique concrete* and hip hop?

Copyright: © 2022 Israel Neuman. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any médium, provided the original author and source are credited.

Along with the introduction of *musique concrete*, Pierre Schaeffer presented a theoretical body of work known as Schaefferian theory. To this day this is the most significant theoretical work on electronic music. Theoretical analysis of hip hop music is scarce. Adams writes about hip hop that it "resists traditional modes of musical analysis more than any other genre" [4]. In one of the main studies of rap music [5], Krims uses the term "music poetics" as a substitution for music theory to stress the need for an analytical method that takes in to account the text, the music and the cultural and social context by which they were created, as well as their creators. Krims analyzes the synchronization of the text with 16-points subdivision of the bar [5]. Studies by Sewell and Boone have suggested typologies as methods for hip hop analysis. Sewell focuses on the compositional functions of samples in sample-based hip-hop songs [6]. Boon presents a typology of works in popular music that are using sampled material [7]. In the following pages we are going to draw on technical and esthetical connections between *musique concrete* and hip hop, and suggest the use of the core typology of Schaefferian theory, the TARTYP, coupled with the TARTYP generative grammars [8, 9] as a method for analyzing hip hop beats.

2. BACKGROUND

A comparison between hip hop and *musique concrete* is based on the fact that both genres are using recorded sound as source material in the compositional process. This is in contrast to using sound recording to capture a performance or a realization of a composition that otherwise exists as an abstract idea expressed in symbolic notation. In regard to hip hop Adams says:

> In hip-hop, the sound object and the composition are one: songs exist as recordings, rarely transcribed into musical notation and even more rarely performed from musical notation [4].

The term sound object is fundamental to Scaefferian theory. Chion defines the sound object as follows:

> The name sound object refers to every sound phenomenon and event perceived as a whole, a coherent entity, and heard by means of reduced listening, which targets it for itself, independently of its origin or its meaning [10].

He continues to say that the "sound object is not a notated symbol on a score" [10]. Both genres gravitate towards the specifics of sound and away from its abstraction and symbolic representation. While Chion also says that "the sound object is not a recorded fragment" [10] because a recorded fragment can be processed to sound in different ways, it is

clear that in both genres recoding techniques are the main tools for defining and fixing the identity of sound objects.

Differences between *musique concrete* and hip hop are found in artistic choices, starting with choices of sound sources. As demonstrated in [11], *musique concrete* considers a wide array of sound sources including studio recordings of traditional acoustic instruments, studio recordings of non-traditional acoustic instruments and techniques such as prepared piano and extended techniques, sound generated by found objects, sound captured in different environments (soundscapes), and electronically generated sound. In the hip-hop golden era of the 1980s and 1990s, sampling was done primarily from existing recordings of popular music in the format of analog LPs (vinyl records) [3]. These recordings were mainly of conventional instruments (acoustic and electric) playing tonal harmonies and melodies. However, their use in hip hop productions was much less conventional. Scratching and turntablism, focusing on the sampling of percussive sounds, introduced noise elements. The blurring of the bass by boosting sub-sonic frequencies, along with the cyclic nature of the music reduced its functionality in the tonal environment.

Starting with DJ Kool Herc in the early 1980s, the practice of sampling was focused on creating "breaks" with excerpts from well-known songs that could be recognized by the audience. Schloss describes the evolution of the deejay's approach to breaks as moving away from the recognizable songs towards the producer's "unique selections." While a canon of recordings suited for breaks was developed and was a prerequisite for hip hop producers, many producers built their reputation based on their ability to find rare recordings to sample their breaks from. Many producers rejected resources such as the Ultimate Breaks and Beats collections to maintain their original choices of sampling [3]. This focus on rare recordings for sampling breaks emphasized the importance of the break itself and how it sounds while diminishing the importance of the source and the audience ability to recognize the original song.

The treatment of the break as a coherent entity, independent of its sampled source, goes even deeper into the hip hop compositional and production process. Through interviews with producers Prince Paul and Jake One, Schloss shows how the unique and concrete characteristics of the sample influence and guide the hip hop producer's compositional choices. Prince Paul maintains that consistency of texture plays an important role in the selection of samples he uses in his beats. Schloss in turn sees Prince Paul's approach as "an aesthetic that is more concerned with a cohesive organizing principle than the diversity of individual elements that fall into its orbit" [3]. He further argues, based on a 1998 interview with Jake One, that hip hop production prefers sampling over live instruments because the very specific (i.e., concrete) character of the sample provides the producer better "musical clues" into the compositional process. In contrast, live instruments, according to Jake One, "could play anything" [3], i.e., interpretations of symbolic music notation performed by a live musician could yield a number of actual sound recordings of which the producer has little control over.

The ideas expressed by hip hop producers, which were discussed so far, are comparable with some of the fundamental concepts of Schaefferian theory and *musique concrete*. Both genres have emerged from the latest innovations of the day, primarily technological innovations, and in spite of strong resistance from established musical genres. Schaeffer writes about *musique concrete*:

> We will look at how we introduce a new element, a present-day invention, in fact, into a somewhat fossilized musical heritage that is, however, nothing other than the summation of successive inventions, going back through history, a whole evolution of musical concepts, always related to the means available to music [12].

Furthermore, two of the basic ideas guiding Schaeffer's concept of reduced listening are also expressed by hip hop producers in the above passages. In reduced listening, the sound object is targeted "for itself independently of its origin or its meaning" [10]. This is similar to deejays realizing that the recognizability of the sampled source is insignificant to the quality of the break. The hearing of the sound object as a coherent unit is tied in Schaefferian theory with identifying the object's sound characteristics as demonstrated by Schaeffer's focus on typology and the TARTYP in particular. Again, a parallel idea is expressed by hip hop producers acknowledging that the sound characteristics of the sample inform the compositional process.

This last comparison is reinforced by both genres' similar approach to music notation. As Adams maintains, hip hop producers do not use music notation [4]. Chion claims that "the sound object is not a notated symbol on a score" [10]. The Schaefferian rejection of music notation emerged as resistance to a contextual environment in which the composition's exitance is first and foremost on the page as music notation. The genres preceding hip hop, such as blues and jazz, already had substantial part of the music done without music notation. But hip hop takes the same principle further to resist also the use of live instruments. As Jake One explains in the interview with Schloss, live instruments "could play anything" [3], i.e., we cannot predict how the performance will sound. The sound object and sampled break on the other hand are concrete recordings with defined characteristics and specific audibility.

3. BEAT CLASSIFICATION

The discussion in this section will focus on the "beat", i.e., the instrumental tracks that accompany the vocals in the hip hop song. These instrumental tracks are usually produced by a combination of tools and techniques including sampling, software instruments, synthesis and to lesser extent live instruments. The hip hop esthetics perceive the song production as a dichotomy of the vocals and the beat. These two entities are usually distinguished by creators as the hip hop artist is responsible for the vocals and the hip hop producer is responsible for the beat. Furthermore, the vocals and the beat may undergo independent creative processes. The artist would many times create the flow (i.e., lyrics) on a pre-existing beat that was partially or fully composed by the producer before any of the song's ideas were conceived.

The methodology that we present in this section is derived from Schaefferian theory and its core typology, the TARTYP[1]. The latter is a table describing sound objects in the time and frequency domains. The sound object types notated at the body of this table are defined by the intersection of the temporal and timbral characteristics specified at its margins. The time domain of the table includes two sets of terms: one describing the duration of sound objects; the other describing the formation of sound objects. The frequency domain of the table includes terminology that identifies four types of timbres.

In [11], Schaeffer provides sound recordings that exemplify each type of TARTYP sound objects, known to most theorists as the *TARTYP sound examples*. While the table does not quantify the different durations of sound objects, by examining the TARTYP sound examples we can identify a time range for each duration type (see below). Timbre and formation are much harder to quantify not only based on Schaeffer's terminology but also based on subjective listening to his examples. In this section, we will use the timbreID tool [13, 14] to generate quantitative timbre and formation classifications of the TARTYP sound examples. We will then use the classifications derived in this fashion to analyze samples of the beat's basic components, mapping between these components and TARTYP sound object types. Such mapping will enable us, in the following section, to use the TARTYP generative grammars [8, 9] in the analysis and re-composition of a representative hip hop drum pattern.

3.1 Hip Hop Production Paradigms

The hip hop instrumentation emerged from the rhythm and blues, funk and soul music of the late 1970s. While contemporary production of beats is often entirely computer based, production software offers virtual recreation of the acoustic, electric and electronic instrumentation of these genres. Production software templates follow arrangements of popular music with a rhythm section of drums, bass, keyboard, and guitar at the base of the arrangement, and additional instruments such as synthesizers, brass instruments, woodwind instruments, and strings, all of which are virtual instruments modeled on acoustic, electric or electronic instruments. Samples of pre-existing recordings are commonly offered as libraries of loops, or collections of sound files containing musical building blocks, such as drum patterns, chord progressions, riffs and licks, that can be repeated (i.e., looped) in different tempos and transpositions while maintain their musical integrity.

The basic components of a beat would fall in to five categories: drum simulations, bass instruments, harmonic progressions, pads, and leads. Drum simulations include drum machines and their virtual recreations, drum samples and drum loops. Bass instruments include virtual recreations of electric and acoustic basses, as well as bass and sub-bass synthesizers, and bass stations. Cyclic harmonic progressions, typical to some hip-hop sub-genres, are performed with a variety of keyboards including pianos, vintage electric pianos and organs (e.g., Fender Roads and Hammond B3), and digital keyboards. To lesser extent, cyclic harmonic progressions are also performed with acoustic and electric guitars. Pads are sustained drones and harmonies generated by synthesizers and "strings" (i.e., electronic or virtual recreations of orchestral string sections). Leads are background or contra melodies mostly generated by synthesizers.

Beats production follows the production and postproduction practices of popular music, including multitrack recording, overdubbing, mixing and mastering. Typical multitrack recording and mixing sessions would include the individual parts of the drum kit recorded on 8 to 12 tracks, bass and keyboards each recorded on 1-2 tracks, one or more tracks of guitars, and one or more tracks of lead and backing vocals. When using drum machines and virtual instruments, the recommendation is to separate these to individual stems (see [15]). Since beats are composed without the use of notation, the DAW software workspace and timeline replace the musical score. Track labeling identifies the instrumentation. Markers on the timeline identify the sections of the songs. In many cases new timbres are generated by software or electronic instruments and are recognizable only by the track labels. At the same time, common track labels like "Synths", "Pads" or "Leads" may say little about the timbre of the instrument and more about the function of this instrument in the arrangement.

The exploration of new timbres is part of the esthetics of beat production. Brett maintains that many producers are interested in creating a new and unique set of sounds for each song they produce [16]. Producer Bryan-Michael Cox recognizes the creation of new sounds as the main way for emerging producers to standout in the crowded field of hip hop production [17]. Layering of multiple timbres, sometimes of the same instrument, is another beat esthetic. For example, the multitrack session of the song *Window* by Killa-L, used here as one of the sources of beat samples, includes several layers of kick drum sounds each only slightly different than the other. Another fundamental esthetic of beats is repetition. The latter may occur in almost every aspect of the arrangement from the basic patterns to a formal organization of repeated sections. Many beats will include repeated chord cycles of 2 to 4 chords, derived from jazz harmony, and used more as colors and less as functional harmony.

3.2 Beat Samples

To create a mapping between the beat's basic components and the TARTYP sound object types, we collected samples (i.e., recorded sound objects) from a number of professionally produced hip hop songs and beats. We took the samples from the pre-mastered multitrack sessions of these songs and beats to allow access to the individual instrumental tracks. The sampled material included complete multitrack sessions of songs by hip hop artists which are available for educational and research uses through the

[1] Detailed discussion of the TARTYP and its structure, as well as a figure of the TARTYP are available in [8, 9].

Mixing Secrets Library[2] [18]. In addition, samples were taken from complete multitrack sessions of beats created by Houston based producer Rickey Benz as well as multitrack sessions produced by the author.

In the process of collecting samples from multitrack sessions, we followed the practices and esthetics of beat production. We identified the instruments and their function in the arrangements based on the multitrack session's track labeling and not by subjective listening to the instrument's timbre. For example, we categorized samples as pads if the track they were taken from was labeled "Pad" even if their timbre resembled the timbre of orchestral strings. The category of "Synth," therefore included a wide variety of timbres. On the other hand, the category of "Kick" was more consistent in timbre.

We determined the durations of samples based on the overall arrangement of the beat as well as the specific schemes of repetition and loops used to create the beat. The musical content of samples varied from a single note to complete phrases or chord progressions. When possible, sampling followed the editing visible in the multitrack session. For example, the samples taken from loops were the clips duplicated to create these loops. Drum samples tended to have short durations between 0.5 to 2 seconds. The drum samples included a single drum or cymbal stroke or a single gesture such as the hi hat roll known as "Trap Hat" [19] or drum fills. In contrast, the durations of samples taken from pads often exceeded 10 seconds. When possible, samples were taken from silence to silence.

The diversity of the hip hop samples in regard to timbre, duration and musical content corresponds to similar diversity displayed by the TARTYP sound examples. In the discussion of the typology of Balanced objects in [11], Schaeffer provides four sets of sound recordings that exemplified the nine Balanced sound objects. The sound sources of these recordings are specified in the example labeling as follows:

- Tracks 31-33 are labeled "instrumental"
- Tracks 34-36 are labeled "from a prepared piano"
- Tracks 37-39 are labeled "concrete"
- Tracks 40-42 are labeled "electronic"

The examples labeled "instrumental" include recordings of trumpets, piano, woodwinds and percussion, i.e., clearly identified instrumental timbres consistent with the label. The timbral identities of the examples labeled "concrete" and "electronic" are much less clear. Within the "concrete" examples one can identify both electronically generated sound (the impulse and iterative) and processed violin and piano sounds (sustain). While the "electronic" examples are all electronically generated they display a variety of timbres. Hence, Schaeffer's use of labels such as "concrete" and "electronic" is similar to the hip hop producers' use of the labels "Synth" and "Pad" where the label is derived from the sound generation method or the musical function but say very little about the timbre.

Schaeffer, in these sound examples, demonstrates the diverse possible manifestations of the sound object as an "event perceived as a whole, a coherent entity" [10]. The variety of timbres, temporal organizations and in particular musical content displayed by these examples demonstrates the sound object as such (a coherent entity), whether it is a single note or a fully composed orchestral passage. Similarly, in the sampling of beat material we aimed to reveal the sound objects within the instrumental tracks of the hip hop song. Therefore, the beat samples display similar diversity. Neither Schaeffer's examples nor the beat samples aim to demonstrate an individual timbral characteristic or an individual temporal characteristic. Nonetheless the identification of sound object types based on the TARTYP is done by intersecting the temporal and timbral characteristics specified at the margins of this table. And therefore, to identify the sound object type of each beat sample we must analyze its temporal and timbral characteristics.

3.3 Quantitative Analysis and Classification

The quantitative analysis we present in this section is focused on measurable characteristics of TARTYP sound objects. Framed within machine learning paradigms, the analysis considers Schaeffer's examples as training datasets and the beat samples as testing datasets. Since the TARTYP includes only descriptive terms of sound objects characteristics, the measurement system we have used in this analysis was derived from the TARTYP sound examples. As mentioned above, the identification of TARTYP sound object types is done by intersecting temporal and timbral characteristics. Each of the sound object types is therefore defined in our analysis by two sets of measurements extracted from the TARTYP examples. By applying the same system to the beat samples and comparing their measurements with those of the TARTYP examples, we can associate beat samples with sound object types.

Within the time domain, the TARTYP applies two sets of descriptors: one for duration and the other for formation (*facture*)[3]. Duration is described by the terms that appear at the top of the table: *short duration, micro-objects*; *moderate duration, temporal unity*; and *long duration (macro-objects) of no temporal unity*. Formation (*facture*) is described by five terms appearing below the duration terminology: *impulse*; *formed iteration*; *formed sustain*; *nonexistent facture*; and *unpredictable facture*. The duration terminology clearly identifies three types of durations: short, moderate and long. The formation (*facture*) terminology is much less clear. However, additional notation at the bottom of the TARTYP provides more clarification. This notation divides the table into *iterative sounds* (the three right columns) and *held sounds* (the three left columns), leaving out the column with the header *impulse* at the center of the table.

It is clear from this TARTYP terminology that sustain and iteration are fundamental elements when defining formations. The latter can be described also as amplitude envelopes where a sustain sound includes a single attack and iterative sound includes multiple attacks. The Schaefferian approach to iteration is explained by Chion in [10]:

[2] The artists and songs are as follows: Killa-L, 'Window'; Little Chicago's Finest, 'My Own'; M.E.R.C. Music, 'Knockout'; Ryan Cali, 'Back Down', 'Crazy'

[3] The terms *form* and *facture* are discussed in [10]. Nonetheless the term *facture* is subject to much sematic debate among Schaefferian scholars.

> The name iterative is given to sounds whose sustainment is prolonged by iteration, i.e., by repetition of impulses at close intervals.

He also maintains the following in [10]:

> Conversely, if the iteration is too spaced out, the iterative sound is no longer perceived as a unit, and each impulse once again becomes an isolated sound object.

Therefore, in our analysis we considered a drum pattern not as a formed iterative sound object but as a combination of impulse sound objects. On the other hand, we considered the trap hat gesture [19], being a series of attacks in very close time intervals, as a formed iterative object. Another example of formed iteration are sounds generated by granular synthesis.

Duration is easily measured in seconds and milliseconds. To define a time range for each one of the three duration types we measured the durations of the TARTYP sound objects associated with this type. Specifically, the sound examples of short duration objects are 1 second or less; the sound examples of medium duration objects are 1-10 seconds; and the sound examples of long duration objects are 10 seconds or more. We then divided the beat samples into three sets, each of which is associated with one of the TARTYP duration types, based on the samples' measured durations.

In measuring formation (*facture*), we focused on identifying the attacks occurring throughout the duration of the sound. Attack detection is done by the bark~ object included in the timbreID tool kit we have used for timbral analysis (see below). The bark~ object is used to trigger spectral analysis and assignment of timbre IDs within timbreID patch [13]. A sound object including a single attack will be assigned one timbre ID. A sound object including multiple attacks, will receive multiple timbre IDs. The use of the timbreID patch for strictly timbre identification as demonstrated in [13] considers only single attack sounds. However, since we are analyzing two dimensional TARTYP sound objects, some of which characterized by *formed iteration*, we must consider multi-attack sound objects. Using attack detection within the timbre identification process allows us to bridge between formation (*facture*) and timbre, characteristics separated in the TARTYP to two different domains. With the timbreID patch, sound object examples and hip hop samples receiving similar timbre IDs will also have a similar formation (*facture*).

We elected to use the machine learning tool timbreID within the Pure Data environment [13, 14] for analysis and measurement of sound objects characteristics in the frequency domain. This tool defines a classification of timbres based on sets of features taken from audio recordings used as the training set. The tool creates a database of timbres in which each timbre receives an ID number. Consequently, the database can be used to identify these timbres in audio recordings used as the testing set. The training set in our analysis was the collection of TARTYP sound examples. The testing set was the collection of samples of the beat's basic components taken from the multitrack recording sessions of hip hop songs, i.e., the beat samples. The goal of this analysis was to point out similarities in timbre between the TARTYP sound examples and beat samples.

Since, as mentioned above, attack detection is incorporated within the timbreID patch, the process of timbre identification is affected by the duration of the audio recording, i.e., the timbreID patch tends to define more timbres in longer audio recordings. To increase the accuracy and effectiveness of the classification, we divided the training set into two subsets, one of short and medium durations and the other of long durations. In the TARTYP, the sub-collection of short and medium durations at the center of the table includes 12 sound object types: 9 Balanced, and 3 Excentric (see [8, 9]). In [11], Schaeffer provides about three sound examples for each sound object type. Hence, the training set of short and medium duration included 35 audio files from Schaeffer's sound object examples. While similar analysis was conducted also in regard to sound objects of long durations the discussion here will focus on the short and medium durations subset.

Sound Object/ Formation	**Num. of Examples**	**Avg. Duration (seconds)**	**Timbre ID range**	**Avg. Attack Detections**
N (sus)	2	6.5	18-20	1.5
X (sus)	3	5.6	67-76	3.3
Y (sus)	3	5	90-96	2.3
W (sus)	4	6	28-48	5.25
N' (imp)	3	2	15-17	1
X' (imp)	3	2.3	64-66	1
Y' (imp)	3	2	87-89	1
Φ (imp)	2	2	21-27	3.5
N" (iter)	3	4.3	6-14	3
X" (iter)	3	4.6	49-63	5
Y" (iter)	3	4.3	77-86	3.3
K (iter)	3	1	0-5	2

Table 1. Timbre IDs associated with each sound object in the database of 96 timbres yield by BFCC analysis (sus = formed sustain; imp = impulse; iter = formed iteration).

The analysis of the short and medium duration training subset was done with 25 features of Bark-frequency cepstrum (BFCC) generated by the bfcc~ object from the timbreID tool kit. The BFCC analysis of the short and medium duration subset yielded a database of 96 timbre IDs. Based on this analysis, each TARTYP sound object type was associated with a range of timbres within the database. Table 1 shows the timbre IDs associated with each sound object in the database of 96 timbres provided by the BFCC analysis. This table also specifies the number of examples used for each sound object type, the average duration of these examples and the average number of attacks detected in these examples. The latter are measurements describing the sound objects' formations which are specified in parenthesis together with the sound object notation.

The training set database was used to identify timbre similarities between the TARTYP sound objects and the beat samples. These samples were categorized based on the labeling of the tracks they originated from and the function of these tracks in the arrangement of the song. The categories of beat samples were: Kick, Snare, Clap, Hi-Hat, Cymbal, Bass, Pads, Synth, and SynthFX. These samples were marked as having either short and medium durations (between 0 to 10 seconds), or having long durations (more than 10 seconds). All sample categories included samples of short and medium duration, although only some categories included samples of long duration. This can be explained by musical functionality and content. For

example, there were no long duration samples in the categories of Kick and Snare, since, in accordance with the TARTYP approach to iteration mention above, these samples included single drum strokes and therefore are generally very short. Long durations appeared only in categories such as Bass, Pads and Synth, as these include sustained harmonies and drones. Table 2 specifies the beat sample categories, the number of samples in each category, and the breakdown by short/medium duration vs long duration.

Category	Total No. of Samples	Short/Medium Duration Samples	Long Duration Samples
Kick	15	15	0
Snare	10	10	0
Clap	6	6	0
Hi Hat	11	11	0
Cymbal	11	11	0
Bass	9	7	2
Pads	12	5	7
Stynth	16	15	1
SynthFX	5	5	0

Table 2. Beat samples

The analysis of beat samples was done in categories. Like the discussion of the training set, the discussion here will focus on samples of short and medium duration. The samples of each category were tested against the BFCC database, yielding varying numbers of timbre IDs associated with the category. The testing results provided a list of timbre IDs that were recognized within the category and the number of times each timbre ID was recognized. The testing results also showed the number of attack detections by the bark~ object for each sample and the average of detections for the category. From a TARTYP perspective, the most commonly occurring timbre ID and average number of attack detections define the timbre and formation of the beat sample category. This information can be compared with the range of timbre IDs and number of attack detections defined by the training database for each sound object and in turn point to associations between beat sample categories and TARTYP sound objects.

For example, the kick category included 15 samples. The average attack detection for this category was 1.13 attacks per sample. The timbre IDs recognized in this category were: 21 (twice), 22 (twice), 35 (once), 37 (once), 38 (twice), 39 (once), 90 (twice), 92 (three times) and 96 (three times). Hence 8 timbre ID occurrences were in the range of the Y sound object (90, 92, 96); 5 timbre ID occurrences were in the range of the W sound object (35, 37, 38, 39); and 4 timbre ID occurrences were in the range of the Φ sound object (21, 22). Based on these timber IDs, the Kick category showed 47% timbral similarity with Y, 29.5 % timbral similarity with W, and 23% timbral similarity with Φ. In regard to formation the average attack detection for the Y sound object is 2.3, the W sound object is 5.25, and for the Φ sound object is 3.5. In addition, out of the three sound examples of the Y object, two examples had only one attack detected while third example had five attacks detected. Hence based on these similarities in duration, formation and timbre the Kick category of hip hop samples can be associated with to the Y sound object. Table 3 specifies attack detections and timbre IDs recognized in the drum-kit categories (Kick, Snare, Clap, Hi hat, and Cymbal) out of the short and medium duration beat samples and their association with sound objects. Table 4 presents the association of short and medium duration beat sample categories with sound objects.

Category	Avg. Attack Detections	Timbre IDs	Primary Sound Objects	Secondary Sound Objects
Kick	1.13	21, 22, 35, 37, 38, 39, 90, 92, 96	Y (47%)	W (29.5%), Φ (23.5%)
Snare	2.6	18, 36, 39, 44, 64, 65	N (58%)	W (23%) X' (19%)
Clap	1	18, 39, 90	N (50%)	Y (33%) W (17%)
Hi Hat	1.18	65, 34	X' (92%)	W (8%)
Cymbal	1.33	18, 34, 65, 76	N (41%) X' (41%)	X (8%) W (8%)

Table 3. Attack detections, timbre IDs and sound object associations in the drum-kit categories

Category	Primary Sound Objects	Secondary Sound Object
Kick	Y (47%)	W (29.5%), Φ (23.5%)
Snare	N (58%)	W (23%), X' (19%)
Clap	N (50%)	Y (33%), W (17%)
Hi Hat	X' (92%)	W (8%)
Cymbal	N (41%), X' (41%)	X (8%), W (8%)
Bass	Y (94%)	W: (4%), Φ: (2%)
Pads	X (39%), Y" (21%)	X' (10.5%), Φ (10.5%) N (5.5%), K (5.5%) W (5.5%), X" (2.5%)
Synth	W (37.5%), N (25%)	K (12.5%), X (6.25%) Y (6.25%), X' (3.125%) X" (3.125%), Y" (3.125%) Φ (3.125%)
SynthFX	Y (50%), Y' (24%)	Φ (13.5%), N (6.5%) W (4%), X" (2%)

Table 4. Associations of beat sample categories with sound objects

4. GENERATIVE GRAMMARS

The results presented in Table 3 and Table 4 show the strong association of the drum-kit categories to three TARTYP sound objects N, Y and X'. These sound objects are members of the Balanced subcollection of the TARTYP [8, 9, 10]. The TARTYP generative grammars, presented in [8] and [9], include grammar definitions for each TARTYP sub-collection and detailed demonstration of rewrite rules in the Balanced subcollection grammar. In this section we will use this grammar to describe and recompose a typical drum pattern which is commonly used in hip hop beats, demonstrating the potential application of the TARTYP generative grammars for beat analysis.

The Balanced generative grammar as presented in [8, 9] includes six terminal symbols, each of which represent a sound object characteristic either in the time or frequency domain of the TARTYP. As such, each terminal symbol is equivalent to a subset of the Balanced object subcollection. Furthermore, each sound object, based on its characteristics, is described by two terminal symbols, one from the time domain and one from frequency domain. Using the association of the drum-kit categories with sound objects presented in Table 4, we can map these categories to terminal symbols, and consequently each category would be

described by two terminal symbols. Equations (1.1) through (1.6) specify the Balanced grammar terminal symbols, their equivalent sound object subsets, and the mapping to drum-kit categories.

DEFINITE = {[N | N′ | N″ |]+} ← { Snare | Clap | Cymbal} (1.1)
COMPLEX = {[X | X′ | X″ |]+} ← { Cymbal | Hi Hat } (1.2)
VARIABLE = {[Y | Y′ | Y″ |]+} ← {Kick} (1.3)
IMPULSE = {[N′ | X′ | Y′ |]+} ← { Cymbal | Hi Hat } (1.4)
FORMED ITER = {[N″ | X″ | Y″ |]+} (1.5)
FORMED SUS = {[N | Xl Yl]+} ← { Snare | Clap | Cymbal | Kick} (1.6)

Figure 1 presents a basic drum pattern that is commonly used in hip hop beats [20]. This pattern utilizes only three drum-kit categories: Kick, Snare and Hi Hat. Equations (2.1) through (2.6) specify the mapping of these three categories to the terminal symbols of the Balanced grammar.

DEFINITE = {[N | N′ | N″ |]+} ← { Snare} (2.1)
COMPLEX = {[X | X′ | X″ |]+} ← { Hi Hat } (2.2)
VARIABLE = {[Y | Y′ | Y″ |]+} ← {kick} (2.3)
IMPULSE = {[N′ | X′ | Y′ |]+} ← { Hi Hat } (2.4)
FORMED ITER = {[N″ | X″ | Y″ |]+} (2.5)
FORMED SUS = {[N | X| Y|]+} ← { Snare | kick} (2.6)

Alternatively, equations (3.1) through (3.3) specify the two terminal symbols describing each one of the drum-kit categories included in the drum pattern of Figure 1.

Kick = FORMED SUS + VARIABLE (3.1)
Snare = FORMED SUS + DEFINITE (3.2)
Hi Hat = IMPULSE + COMPLEX (3.3)

Based on the Balanced grammar definition from [8, 9] and the relationships specified in equations (2.1) through (2.6) as well equations (3.1) through (3.3) we can now define legal sets of rewrite rules within the Balanced grammar that can in turn generate rewrite paths describing drum patterns including a kick, snare and hi hat, such as the one shown in figure 1. Equations (4.1) through (4.14) present such a rule set with the start symbol balanced:

balanced → FORMED SUS bal_expre_v (4.1)
balanced → IMPULSE bal_expre_c (4.2)
bal_expre_v → VARIABLE FORMED_SUS (4.3)
bal_expre_v → VARIABLE FORMED_SUS bal_expre_i (4.4)
bal_expre_v → VARIABLE FORMED_SUS bal_expre_fs (4.5)
bal_expre_i → IMPULSE bal_expre_c (4.6)
bal_expre_c → COMPLEX IMPULSE (4.7)
bal_expre_c → COMPLEX IMPULSE bal_expre_fs (4.8)
bal_expre_fs → FORMED_SUS bal_expre_d (4.9)
bal_expre_fs → FORMED_SUS bal_expre_v (4.10)
bal_expre_d → DEFINITE FORMED_SUS (4.11)
bal_expre_d → DEFINITE FORMED_SUS bal_expre_fs (4.12)
bal_expre_d → DEFINITE FORMED_SUS bal_expre_i (4.13)
bal_expre_c → COMPLEX IMPULSE bal_expre_i (4.14)

This rule set is legal within the Balanced grammar since it contains two rules with start head balanced, it contains no duplicate rules, and all non-terminal symbols that appear in the body of a rule have at least one other corresponding rule where they appear only in the head. This rule set is useful for describing the drum pattern since all terminating rules include a pair of terminal symbols matching one of the drum-kit categories as specified in equations (3.1) through (3.3). In addition, the set includes non-terminating rules with pairs of terminal symbols matching the three drum-kit categories discussed. Therefore, a path generated in this rule set will include one or more pairs of terminal symbols matching one or more of the three drum-kit categories.

The path presented in equation (5) was generated by the rule set of equations (4.1) through (4.14).

Path: → R:4.1 → FORMED_SUS → R:4.4 → VARIABLE FORMED_SUS (= Kick) → R:4.6 → IMPULSE → R:4.14 → COMPLEX IMPULSE (= Hi Hat) → R:4.6 → IMPULSE → R:4.8 → COMPLEX IMPULSE (= Hi Hat) → R:4.10 → FORMED_SUS → R:4.13 → DEFINITE FORMED_SUS (= snare) → R:4.6 → IMPULSE -→ R:4.14 → COMPLEX IMPULSE (= Hi Hat) → R:4.6 → IMPULSE → R:4.7 → COMPLEX IMPULSE (= Hi Hat) (5)

It describes the first two beats of the pattern in figure 1 including the first kick strike, the first two hi hat strikes, the first snare strike, and the third and fourth hi hat strikes. The regeneration of these two beats using the path of equation (5) is described in figure 2.

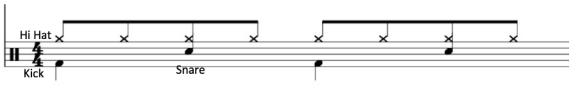


Figure 1. Basic hip hop drum pattern

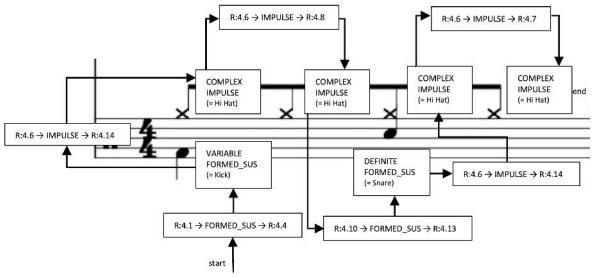


Figure 2. Regeneration of the first two beats of the pattern in figure 1 using the path of equation (5).

5. CONCLUSIONS

The clear association between the drum-kit categories and the Balanced sound objects, as presented in tables 3 and 4 enabled us to use the TARTYP generative grammars in the description of a basic drum pattern. At the same time, our classification of other beat sample categories have shown much less decisive association with TARTYP sound objects. This can be explained by the diversified makeup of these categories as opposed to the more uniform makeup of the drum-kit categories. For example, the Pads category is defined by the function of the track within the arrangement and may include a variety of timbres from orchestral strings to unidentified Sci-Fi sounds. This is in contrast with the drum-kit categories in which sounds more closely resemble the acoustic instrument.

In future research we suggest using the methods demonstrated in this paper in the analysis of beats produced by the same producer. As noted by [3] and [17] hip-hop producers tend to develop their sound identity based on the selection of instruments, samples and sounds they use in

the production of their beats. Therefore, a classification of beat samples taken from multiple beats produced by the same producer is likely to yield more decisive association of all sample categories with TARTYP sound objects. This in turn will allow us to use rewrite rules in the TARTYP generative grammars for analysis and re-composition of the complete beat arrangement.

6. REFERENCES

[1] S. Howell, "The Lost Art of Sampling: Part 1" *Sound on Sound* 2005 [Online]. Available: https://www.soundonsound.com/techniques/lost-art-sampling-part-1 (accessed January 1, 2022).

[2] T. Tully, 'Choosing the Right Sampler', *Electronic Musician* (1986), pp. 27-34.

[3] J. G. Schloss, *Making Beats: The Art of Sample-Based Hip-Hop*. Wesleyan University Press, 2014.

[4] K. Adams, "The Musical Analysis of Hip-hop" in *The Cambridge Companion of Hip Hop*, J. A. Williams ed. Cambridge University Press 2015 pp. 132-148.

[5] A. Krims, *Rap Music and the Poetics of Identity*. Cambridge University Press, 2000.

[6] A. Sewell, "A Typology of Sampling in Hip-Hop," Ph.D. dissertation. Indiana University, 2013.

[7] C. Boone, "Mashing: Toward a Typology of Recycled Music," *Music Theory Online* 19/3 (2013). Available: http://mtosmt.org/issues/mto.13.19.3/mto.13.19.3.boone.php (accessed January 1, 2022).

[8] I. Neuman, "Generative Grammars for Interactive Composition Based on Schaeffer's TARTYP," *Proceedings of the International Computer Music Conference*, Perth, 2013.

[9] I. Neuman, "Generative Tools for Interactive Composition: Real-Time Musical Structures Based on Schaeffer's TARTYP and on Klumpenhouwer Networks," *Computer Music Journal* 38 no. 2 (2014):63-77.

[10] M. Chion, *Guide to Sound Objects: Pierre Schaeffer and Musical Research*. J. Dack and C. North, trans. Buchet/Chastel, (1983) 2009.

[11] P. Schaeffer, *Solfège de l'objet sonore*, sound recording. Paris, INA-GRM, 1998

[12] P. Schaeffer, *Traité des objets musicaux*. Paris, Éditions du Seuil, 1966.

[13] W. Brent, "A Timbre Analysis And Classification Toolkit For Pure Data," *Center for Research in Computing and the Arts* University of California, San Diego, 2009. [Online]. Available: http://williambrent.Conflations.com/ papers /timbreID.pdf (accessed January 9, 2015).

[14] W. Brent, "Cepstral analysis tools for percussive timbre identification," in *Proceedings of the 3rd International Pure Data Convention*, Sa ˜o Paulo, Brazil, 2009.

[15] M. Senior, *Mixing Secrets for the Small Studio*, 2nd Edition. Routledge, 2018.

[16] T. Brett, "On the Limits of Musical Timbre," *Brettworks: Thinking Through Music*. Available: https://brettworks.com/2019/03/18/on-the-limits-of-musical-timbre/ (accessed January 1, 2022).

[17] Moog Music Inc., *SOUNDCRAFTS | Bryan-Michael Cox*, YouTube, Aug 23, 2013 [Video file]. Available: https://www.youtube.com/watch?v=eT1M13-1Fgc (accessed January 1, 2022).

[18] M. Senior, "The 'Mixing Secrets' Free Multitrack Download Library," *Cambridge Music Technology*. Available: https://www.cambridge-mt.com/ms/mtk/#HipHop (accessed January 1, 2022).

[19] A. Lavoie, "Trap Hats: 4 Hi-Hat Techniques Every Trap Producer Uses," *LANDR*. Available: https://blog.landr.com/trap-hats (accessed January 1, 2022).

[20] Rudemuzik, "4 Classic Hip Hop Drum Patterns," Feb. 2020. Available: https://rudemuzik.com/blogs/resources/4-drum-patterns-for-hiphop (accessed January 1, 2022).

SMC-22 Paper Session 8

Towards an FPGA-Based Compilation Flow for Ultra-Low Latency Audio Signal Processing

Maxime Popoff,[a] Romain Michon,[b] Tanguy Risset,[a] Yann Orlarey,[c] and Stéphane Letz[c]
[a]Univ Lyon, INSA Lyon, Inria, CITI, EA3720, 69621 Villeurbanne, France
[b]Univ Lyon, Inria, INSA Lyon, CITI, EA3720, 69621 Villeurbanne, France
[c]Univ Lyon, GRAME-CNCM, INSA Lyon, Inria, CITI, EA3720, 69621 Villeurbanne, France
maxime.popoff@insa-lyon.fr

ABSTRACT

Field Programmable Gate Arrays (FPGAs) have been increasingly used in recent years for real-time audio Digital Signal Processing (DSP) applications. They provide unparalleled audio latency and processing power performances. They can target extremely high audio sampling rates and their large number of General Purpose Inputs and Outputs (GPIOs) make them particularly adapted to the development of large scale systems with an extended number of analog audio inputs and outputs. On the other hand, programming them is extremely complex and out of reach to non-specialized engineers as well as to most people in the audio community. In this paper, we introduce a comprehensive FPGA-based environment for real-time audio DSP programmable at a high level with the FAUST programming language. Our system reaches unequaled latency performances (11 µs round-trip) and can be easily controlled using both a software graphical user interface and a dedicated hardware controller taking the form of a sister board for our FPGA board. The implementation of the system is described in details and its performances are evaluated. Directions for future work and potential applications are presented as well.

1. INTRODUCTION

Audio Digital Signal Processing (DSP) has been widely studied and implemented on a wide range of computer architectures: Von Neuman CPUs, multi-cores, GPUs, dedicated circuits, FPGAs, [1] etc., with throughput (or computing power) as the main target, in most cases. However, achieving low audio latency on such systems is an important objective too that is often limited by hardware or the use of an Operating System (OS). For instance, on conventional computers (i.e., "PCs"), samples must be grouped in buffers in order to "hide" from the audio codec chip [2] potential latency induced by interruptions, etc.

The only solution to fully get rid of buffering on digital audio systems – and hence reach the lowest possible latency – is to use a dedicated circuit (i.e., ASIC [3] or FPGA) directly connected to an audio codec. The benefits of using such systems for real-time audio DSP expand far beyond the scope of audio latency by: (i) potentially providing more computational power and throughput, (ii) allowing for the use of very high audio sampling rates (>10MHz), (iii) providing a large number of audio inputs and outputs. However, using FPGAs makes the implementation of DSP algorithms extremely complicated because hardware design [4] is a long and tedious process. In this context, the use of FPGA or VLSI [5] design tools – which are high-added-value proprietary environments – cannot be avoided. However, there has been a trend in recent years towards making them more accessible, which primarily translated into the rise of High Level Synthesis (HLS) for non-professional hardware designers.

In this paper, we propose to simplify the implementation of audio DSP programs on FPGA platforms by providing a fully automated *compilation flow* based on a high-level audio DSP language: FAUST [1]. The FAUST compiler is used in conjunction with HLS tools to program the system down-to hardware. This compilation flow is optimized for audio latency and is able to reach unparalleled figures: 11 µs from analog input to analog output. We demonstrate the use of our tool-chain on the Xilinx Zybo Z7 board connected to various audio codecs. We also introduce a hardware control interface taking the form of a sister board for the Zybo Z7 to facilitate the control of audio DSP on the FPGA.

The actual added values of our work lie in (i) the use of HLS and (ii) solving the most important technical obstacles related to the automatic translation of an audio DSP language to an FPGA, namely: software control of the DSP, initialization of constants, and minimization of external memory accesses.

[1] *Field Programmable Gate Array*

[2] In this paper, "audio codec" always refer to a hardware component providing analog outputs and inputs (audio ADC/DAC): not an audio compression algorithm.

[3] *Application Specific Integrated Circuit*

[4] In the world of FPGA, the term "hardware" can be ambiguous and actually refers to what is happening on an FPGA chip. Hence, "hardware design" here is equivalent to "FPGA programming." This terminology will be used throughout the paper.

[5] *Very Large Scale Integration*

Copyright: © 2022 Maxime Popoff,[a] Romain Michon,[b] Tanguy Risset,[a] Yann Orlarey,[c] and Stéphane Letz[c] et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. STATE OF THE ART

2.1 FPGA and Audio DSP

Audio on FPGA has been studied for a long time with initial contributions in the 2000s [2–4] and more recently [5–7], to only mention a few. Most of these works consist in manually designing a hardware/software implementation of a particular audio process. FPGA design was so complex that these implementations were actually published as research papers. In 2014, Verstraelen proposed an innovative dedicated architecture synthesized [6] on FPGA that could be programmed from data-flow graphs of audio applications [8], but this is not an active project anymore.

In the industry, an increasing number of products are using FPGAs for real-time audio signal processing tasks such as the Novation Summit [7] or some Antelope Audio [8] products (non-exhaustive list).

FPGA design methodologies are evolving quickly and it is now possible to derive hardware description from a high-level specification. High Level Synthesis is now proposed by all FPGA vendors and can be seen as a way to increase hardware designers productivity. It consists in *compiling* a high-level language (usually restricted syntax of C, C++, or Python) down-to hardware. Examples of such tools are Xilinx's `Vitis_HLS`, Mentor Graphic's `Catapult-C` or Intel/Altera's `HLS compiler`. The term *compilation* is a bit reductive because HLS transforms a sequential program into a functionally equivalent circuit containing many parallel computations, opening the door to much more computational power.

IP-based design is another approach proposed by MathWorks HDL coder [9] or Xilinx XSG. [10] A recent study at Lund University [9] compared MathWork HDL coder with vendor tools, but did not extend the comparison to HLS tools. This evolution opened the way to a new category of papers on FPGA audio starting from higher level specifications of audio DSP, greatly facilitating the overall engineering process.

The work of Vannoy et al. [10, 11] is one of the most advanced in this category. They use MathWork HDL coder and an Intel Cyclone V FPGA. Their complete design is available on GitHub and they provide many additional facilities (i.e., Linux sound card driver, web client for control interface, etc.). An interesting work on blind sailing was proposed by Singhani and Morrow [12]. They mapped "by hand" a Head-Related Transfer Function (HRTF) on an FPGA to achieve low latency performances (with a latency below 5ms as an aim). LowLAG [13] and [14] are other examples of these approaches.

Our FAUST2FPGA compilation flow is implemented in the SyFaLa toolchain [11] [15]. Our system is the only one using HLS and providing an actual *compilation* process. Other works all rely on existing Matlab/Simulink blocks or FPGA Vendors blocks for their design. Additionally, we are the only ones providing support for the automatic use of external DDR memory when Block Rams are not sufficient. Finally, we are also proposing a hardware interface to control audio DSP on the FPGA (see §4).

2.2 Low Audio Latency

The music and audio technology community has been focusing on audio latency for a very long time. Wang's PhD thesis [16] provides a good review of the different potential sources of delay in digital audio systems. For instance, ADC/DAC, [12] operating systems, and audio networking are usually identified as the main contributors to audio latency.

When using FPGAs, the only potential source of latency comes from the ADC/DAC because no operating system is involved to process the sample stream. Most modern audio codecs on the market are based on $\Delta\Sigma$ ADC/DACs. $\Delta\Sigma$ DACs imply the use of filters for signal reconstruction which are the main source of latency on this kind DACs. The codecs that we decided to use are presented in §4.1 and are specifically optimized for latency, mostly by using advanced reconstruction filter designs implying a minimal number of taps (as opposed to more basic codecs that typically use high order digital filters). There is a clear lack of accurate delay reporting mechanisms for $\Delta\Sigma$ ADC/DAC so the effective delay of a given system usually needs to be evaluated experimentally as shown in §4.1.

Among the aforementioned works, the smallest announced latency on FPGA processing, from analog input to analog output is $180\,\mu s$ [10]. In §5.1, we demonstrate how to further reduce latency by a factor of 10.

3. AUDIO DSP TO FPGA COMPILATION PRINCIPLES

Audio DSP programs have a specific structure which can be better expressed in *Domain Specific Languages* (DSLs) such as CSound [17], FAUST, or PureData [18]. The efficient compilation of audio DSP programs to hardware must take this particular structure into account. This paper presents a compilation flow of FAUST programs, but the basic principles used here can be adapted to other DSLs.

This section gives an overview of the structure of audio DSP programs and how it is handled for efficient compilation to FPGA.

3.1 Audio DSP to FPGA Compiler Overview

Figure 1 presents an overview of the FAUST2FPGA compilation flow. This compilation flow has been tested on the Xilinx Zynq-7010 SoC, [13] which is used on the Zybo Z7 board. [14] A FAUST source program – say a filter-based sine wave oscillator such as the one presented on

[6] The term "synthesis" here refers to the process of turning a program specified in a Hardware Description Language (HDL) into a bitstream (as opposed to "sound synthesis").

[7] `https://novationmusic.com/en/synths/summit` (All URLs in this paper were verified on Jan 25, 2022)

[8] `https://en.antelopeaudio.com/`

[9] `https://www.mathworks.com/products/hdl-coder.html`

[10] `https://www.xilinx.com/products/design-tools/vitis/vitis-model-composer.html`

[11] `https://github.com/inria-emeraude/syfala`

[12] *Analog to Digital Converter // Digital to Analog Converter*

[13] *System on a Chip*

[14] `https://www.xilinx.com/products/boards-and-kits.html`

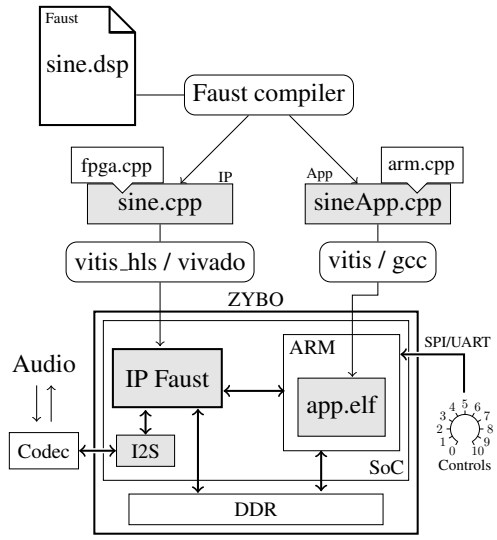


Figure 1. FAUST to FPGA compilation flow, gray boxes are generated during the compilation flow

Fig. 2 (taken from the FAUST libraries) – is compiled to C++, using a specific `fpga.cpp` architecture file and the `-os2` compiler option that was specifically implemented as part of this project (see §3.3). HLS and synthesis tools (`Vitis_HLS` and `Vivado` in our case as we target Xilinx hardware) are used to generate the FAUST IP [15] from the C++ code. The portion of the code associated to control (i.e., the `freq` slider on Fig. 2) is compiled to another C++ program using a specific `arm.cpp` architecture file, to be executed by the ARM processor of the Zynq SoC, which in turn can potentially be interfaced with software or hardware physical controllers (see §4.2). The constants initialization and computations depending on control variables (variables `th`, `c`, and `s` in the `with` region on Fig. 2) are also executed on the ARM processor. The way control is managed in our system is detailed in §4.2. DSP computation – i.e., `nlf2` which implements a second order normalized digital waveguide resonator on Fig. 2 – is executed for each sample on the FPGA by the FAUST IP.

```
import("stdfaust.lib");

freq = hslider("freq [knob:1]",440,50,1000,0.01);
nlf2(f,r,x) = ((_<:_,_),(_<:_,_) :
                (*(s),*(c),*(c),*(0-s)) :>
                (*(r),+(x))) ~ cross
with {
  th = 2*ma.PI*f/ma.SR;
  c = cos(th);
  s = sin(th);
  cross = _,_ <: !,_,_,!;
};

impulse = 1-1';
process =  impulse : nlf2(freq,1) : !,_  <: _,_;
```

Figure 2. Filter-based sine wave oscillator in FAUST illustrating the compilation process.

The FAUST IP is connected to an I2S IP which interfaces the FPGA with an audio codec. The sample rate and the number of audio Input/Output can be configured before compilation.

[15] *Intellectual Property*

3.2 Audio Objects Topology

3.2.1 General Structure of An Audio Program

In audio DSP programs, it is common to distinguish two different rates: the *audio rate*, which is typically between 20 kHz and 192 kHz, and the *control rate*. The control rate corresponds to the rate at which external controllers are updated (typically between 100 Hz and 1000 Hz). As an example, all computations influenced by the `freq` controller in Fig. 2 can be executed at control rate. Conversely, computations for the `nlf2` function should be executed at audio rate.

Another characteristics of audio programs is that some computations are done only once at initialization time. They correspond to constant computations which cannot be done statically because they depend on values known at execution time, such as the sample rate. Similarly, static arrays are heavily used in audio DSP programs, either for waveform tables or for delay lines (e.g., echos, reverbs, etc.), translating into large memory footprints (often requiring dozens of MB of memory).

3.2.2 Adapting Audio Programs to FPGA Architectures

A wide range of FPGA-based boards (such as the Digilent Zybo Z7 used here) integrate an ARM CPU which can be used in conjunction with the FPGA. In the context of audio DSP, FPGA resources shall only be used when needed. Hence, some portion of a program can be ran on the ARM processor, and the rest can be processed by the FPGA itself. As mentioned before, the use of an FPGA for audio processing is potentially beneficial to (i) performances, (ii) throughput of the audio program, (iii) using extremely high sampling rates, (iv) getting a large number of audio input and output channels, or (v) for latency reasons.

Constants initialization (i.e., waveform tables, delays, etc.) does not need to be computed by the FPGA and can be carried out on the ARM CPU before starting DSP. While the available memory on the FPGA itself (called *Block Rams*) is rather limited (about 1 MB on Zynq-7010 SoCs), external memory (DDR) is huge, but accessing it can be fifty times slower than Block Rams. As a consequence, Block Rams should be used in priority by objects used in current sample computations to avoid starvation due to high memory latency. On the other hand, multiple memory accesses to the same array element (to the same sample, for instance) should be cached on the FPGA. This is also true on a GPP [16] where several accesses to the same sample should be carried out through a scalar temporary variable that can be cached in a register by the compiler.

3.2.3 Automatizing This Process With the Faust Compiler

The FAUST compiler typing system is able to identify computations that can be done at *compilation time*, *initialization time*, *control rate*, and finally *sample rate*. Thanks to that infrastructure, the aforementioned dispatching can be done easily:

- Constant initialization should be dispatched on the ARM CPU, then copied on the FPGA.

[16] *General Purpose Processor*

```
[....]
void controlmydsp(mydsp* dsp, int* iControl, float* fControl,
        int* iZone, float* fZone) {
    fControl[0] = (dsp->fConst0 * (float)dsp->fHslider0);
    fControl[1] = sinf(fControl[0]);
    fControl[2] = cosf(fControl[0]);
}
[....]
void computemydsp(mydsp* dsp, FAUSTFLOAT* inputs,
        FAUSTFLOAT* outputs, int* iControl, float* fControl,
        int* iZone, float* fZone) {
    dsp->iVec0[(dsp->IOTA0 & 1)] = 1;
    float fTemp0 = dsp->fRec1[((dsp->IOTA0 - 1) & 1)];
    float fTemp1 = dsp->fRec0[((dsp->IOTA0 - 1) & 1)];
    dsp->fRec0[(dsp->IOTA0 & 1)] = ((fControl[1]*fTemp0) +
        (fControl[2] * fTemp1));
    dsp->fRec1[(dsp->IOTA0 & 1)] = (((float)(1 -
        dsp->iVec0[((dsp->IOTA0 - 1) & 1)]) + (fControl[2] *
        fTemp0)) - (fControl[1] * fTemp1));
    float fTemp2 = dsp->fRec1[((dsp->IOTA0 - 0) & 1)];
    outputs[0] = (FAUSTFLOAT)fTemp2;
    outputs[1] = (FAUSTFLOAT)fTemp2;
    dsp->IOTA0 = (dsp->IOTA0 + 1);
}
[....]
```

Figure 3. Excerpt of C++ code generated by the FAUST compiler from the FAUST code presented on Fig. 2 when tuned for the FPGA target.

- Control rate computations should be dispatched on the ARM CPU and the resulting values copied on the FPGA at control rate.
- Other computations, i.e., sample rate computations, should be dispatched on the FPGA.
- Small objects should be stored on Block Rams as much as possible.
- Large arrays should be stored in external DDR. The caching mechanism is already implemented in the code generated by the compiler as shown by the `fTemp` variables on Fig. 3.

3.3 New -os2 Code Generation Option

A new *-os2* compilation option has been added to the FAUST compiler for the C and C++ backends. Fig. 3 shows an excerpt of the C++ code generated from the code on Fig. 2. It allows for a better separation between control rate and sample rate computations as well as for a better control of the DSP memory layout:

- A separate `controlmydsp` function is generated. It contains the code that will be executed at control rate using the new value of all controllers (i.e., buttons, sliders, etc.). This code is then used to update the FAUST IP internal state. The `controlmydsp` function is called on the ARM side (i.e., right side on Fig. 1).
- The `computemydsp` function is used to compute a single sample (see Fig. 3). It does not contain the loop that is generated in the standard version of the FAUST compiler which would normally process a buffer of input/output samples. Hence, the name of the new option: `-os2` stands for *one sample, version 2*. This function is called on the FPGA side.
- Two additional `iControl` and `fControl` arrays are used as parameters to share the control state between the `controlmydsp` and `computemydsp` functions.
- DSP memory is divided between a section kept in the DSP class/structure, and a separated section typically allocated on the DDR using `iZone` and `fZone` arrays. [17] The DDR will typically contain long delay lines. The user can tune an additional environment variable `FAUST_MAX_SIZE` to precisely control how the memory layout is divided, depending on the amount of memory available in Block Rams.
- Two new functions to copy the constants computed on the ARM side (i.e., in `iZone` or `fZone`) on the FPGA are also generated.

As a summary, these concepts are illustrated by the C++ code excerpt in Fig. 3 generated by the FAUST compiler with the `-os2` option from the DSP program presented in Fig. 2. The `controlmydsp` function is executed on the ARM CPU (`app.elf` in Fig. 1), waiving the need for a floating point Sine and Cosine functions on the FPGA. The portion of the code synthesized on the FPGA is contained in the `computemydsp` function. Caching of duplicated memory accesses can be seen via the `ftemp0` and `ftemp1` variables and exploited by the HLS compiler. In this particular example, there are no large arrays so all variables are stored in Block Rams. Note that the same `-os2` option is used on both the FPGA and ARM sides to preserve the structure of generated code, only the architecture files change (`fpga.cpp` and `arm.cpp` on Fig. 1).

3.4 Faust to FPGA in Practice

As mentioned earlier, FAUST2FPGA is implemented as part of the Syfala compiler [15], which currently only runs on Linux. It is freely accessible on the Syfala GitHub. [18] Installing the Xilinx Tools Suite is a requirement. The control application (`app.elf` on Fig. 1) runs *bare metal* (i.e., without an operating system) on the ARM CPU.

The different tools are called using `make`. Xilinx tools are configured with TCL scripts. Compiling a program such as the one on Fig. 2 takes about fifteen minutes on a standard PC. A big chunk of the time is spent on the Vivado logic synthesis.

4. CONTROLLERS & INTERFACING THE FAUST IP TO THE CODEC

The previous section presented how the FAUST IP is generated from the FAUST code. IP interfacing has to be treated with care because it has an important impact on performances and latency. For instance, in our initial attempt [15] we have shown that a 800 µs latency was added by the built-in audio codec (Analog Devices SSM2603) of the Zybo board, which was not optimized for latency. Section 4.1 explains how we have interfaced the FAUST IP with various codecs in order to obtain the smallest possible latency (11 µs from analog input to analog output). Then Section 4.2 presents the interface for controlling the audio DSP in hardware or in software.

[17] iControl/iZone contain integer values, fControl/fZone contain float values.

[18] `https://github.com/inria-emeraude/syfala`

4.1 Audio Codec Interface

In the current version of the compiler, the FAUST IP has a constant latency of one sample. Hence, only two potential sources of latency can be optimized: the audio codec itself and the serial audio interface (I2S) between the codec and the FAUST IP. Audio codecs usually combine an ADC and a DAC, a serial I2S interface, and some analog filters for signal reconstruction (see Fig. 5). Some codecs also include a hardware DSP (an actual hardware component). In our case, it is only used for signal routing and hence is not a source of additional latency.

In the following paragraphs, the I2S clock rate (which is also the sample rate of our system) is noted as $f_s^{I^2S}$, and the ADC/DAC internal clock rate as Ck^{ADC}. [19] Both can be configured but there are some constraints, in particular on their maximum value.

Three different audio codecs were used for our experiments (very few codecs on the market are optimized for latency and the ADAU1777 and the ADAU1787 are the ones providing the best performances that we could find):

- The Analog Devices SSM2603 (the built-in codec of the Zybo board), which is not optimized for low latency. ($\max f_s^{I^2S} = \max Ck^{ADC} = 96\,\text{kHz}$)
- The Analog Devices ADAU1777, which is optimized for low latency. ($\max f_s^{I^2S} = 192\,\text{kHz}$ & $\max Ck^{ADC} = 768\,\text{kHz}$)
- The Analog Devices ADAU1787, which is the fastest audio codec we used and that provides the best performances in terms of latency. ($\max f_s^{I^2S} = \max Ck^{ADC} = 768\,\text{kHz}$)

The ADC/DAC introduces at least a 1 sample delay. Beyond that, audio latency is directly connected to the sampling rate: the higher the sampling rate, the lower the latency. In practice though, the codec actually introduces more than just one sample of delay. Audio codecs refer to this as *group delay* and corresponds to the delay added by the ADC/DAC. Group delay depends on many parameters, including the sampling rate. For the ADAU1787, the best announced group delay is $5\,\mu\text{s}$ (analog input to analog output), obtained at $Ck^{ADC} = f_s^{I^2S} = 768\,\text{kHz}$. We validated this figure by measuring a $5.9\,\mu\text{s}$ latency when bypassing the I2S, which approximately corresponds to a 4 samples delay (see Section 5.1).

However, this latency does not include the serial audio interface delay. The I2S protocol is clocked with a *bit clock* (Bclk) which depends on the sample rate $f_s^{I^2S}$. The relation between $f_s^{I^2S}$ and f^{Bclk} depends on the bit depth of the audio samples: $f^{Bclk} = f_s^{I^2S} \times 2 \times \text{bit depth}$. Hence, for the ADAU1787, the maximum authorized I2S sample rate is higher with audio samples of 16 bits than for 24 or 32 bits.

Increasing the sampling rate $f_s^{I^2S}$ lowers the latency introduced by the I2S. On the other hand, it decreases the available time for the FAUST IP to perform its computation. So increasing the I2S sampling rate must be done carefully. However, the ADC clock rate Ck^{ADC} can be increased without changing the I2S sampling rate, hence many combinations are possible as explained in Section 5.1.

Once the codec parameters have been optimized, the serial audio interface on the SoC side needs to be managed. To ensure that the latency of the I2S interface is as low as possible, we implemented our own I2S transceiver IP. Prior works [13] use the AXI4-Lite interface but our custom VHDL IP provides us a better control over latency. Our implementation is based on an IP from the digikey forum [20] which was extended to enable arbitrary sample bit depths. Another advantage of this custom I2S IP is that it allows us to adjust the number of channels (audio inputs or outputs). We can therefore produce as many I2S channels as we want – all sharing the same bit clock (Bclk) and Word Select signal – minimizing GPIOs usage to two pins per I2S.

The effective measurements of the latency introduced by the codec interface is presented in Section 5.1. Next section presents the interface used to control the FAUST audio DSP.

4.2 Control Interface

The FAUST IP can be controlled using a physical (i.e., knobs, potentiometers, buttons, etc.) or a software (i.e., running on a computer connected to the FPGA board) interface system which were designed as part our project. They both interact with the FAUST IP through the ARM processor of the FPGA board. When the state of a FAUST DSP parameter is updated on the ARM (whether it came from a software or a hardware action), the updated control state is transferred on the FPGA using the ARM s-axi calling the `sendControlToFPGA` function (see §3.2). This function sends all the new controller states to the FPGA taking a *best effort* strategy: the ARM is infinitely looping and sending new controllers values as fast as it can.

4.2.1 Hardware Interface

The hardware interface is based on a PCB board [21] ("sister board/shield") which can be mounted on top the Zybo Z7 (see Fig. 4). This board is generic enough so that a wide range of combinations of controllers (i.e., buttons, rotary potentiometers, faders, etc.) can explored and soldered on it. It is interfaced to the ARM processor via an ADC chip mounted directly on the board and using the SPI protocol. The physical controllers of the board can be bound to DSP parameters directly in the FAUST program using using metadata such as `[knob:1]` or `[switch:1]`. For example, the `freq` slider of Fig. 2 is bound to the first knob of the hardware interface board, etc. A detailed documentation of this system is available on the SyFaLa GitHub.[18]

4.2.2 Software Interface

The software interface can be automatically generated by the FAUST2FPGA compilation chain (see §3). It is a GTK-based GUI directly adapted from the FAUST architectures environment and that is meant to be executed

[19] Here we will consider that the ADC and the DAC have the same sample rate.

[20] `https://forum.digikey.com/t/i2s-transceiver-vhdl/12845`

[21] The source files of the board are available on the GitHub of the project.[18]

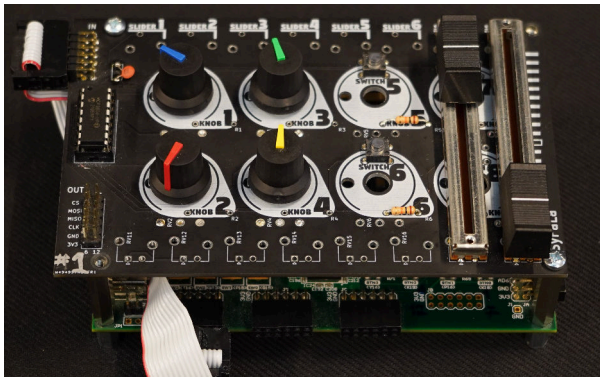


Figure 4. Zybo Z7 with our custom sister board presenting a possible combination of controllers.

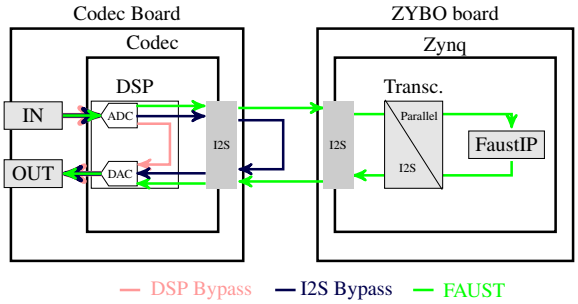


Figure 5. Latency measurements testbench used for the results presented in Table 1. Latency was measured from analog input to analog output using square signals in three different configurations.

on a host computer connected to the Zybo board. Once launched, it communicates with the ARM processor of the Zybo using the UART serial protocol. The corresponding communication classes `UartReceiverUI` and `UartSenderUI` are automatically integrated to the application program and can be used for that purpose. Unlike the hardware controller presented in the previous section, the generated software interface contains all the parameters declared in the FAUST program.

5. RESULTS AND EVALUATION

In this section, we first evaluate the audio latency obtained by our tool-chain. We then measure the impact of our optimizations on memory organization and computations dispatching.

5.1 Latency Performances

Our experimental setup for measuring latency is described in Fig. 5. A 440 Hz square wave signal is sent to a simple FAUST IP implementing a pass through. The latency between the analog input and output of the system is measured by looking at the time difference between zero crossings of the source and of the processed signal. We performed a large number of measurements of this kind. Their standard deviation is always less than one sample.

Three types of measurements were performed for each codec (see Fig. 5) from analog input to analog output:

- a complete pass-through going to the FAUST IP through I2S, as described above (path labeled FAUST in Fig. 5),
- a pass-through going through I2S only (without going through the FAUST IP), this path is labeled "I2S Bypass" in Fig. 5,
- an internal pass-through in the codec (path labeled DSP Bypass), hence not using I2S.

Table 1 shows the latency measurements obtained on various codecs with different sampling rate configurations. The first column indicates the global sampling rate of the system $f_s^{I^2S}$ (the sampling rate used by the I2S interface). The second column indicates the internal sampling rate of the codec: Ck^{ADC} (which therefore can be different from the global sampling rate of the system). As expected, the best latency (11 µs) is obtained with the fastest internal ADC/DAC with an I2S rate of 768 kHz and using 16 bits samples. In the current version of the compiler, the FAUST IP will always add a 1 sample delay. To our knowledge, none of the previous works could achieve such a low latency (i.e., 180 µs was announced in [10] with a codec clock at 768 kHz). Our custom I2S transceiver also adds an additional sample of delay, so the whole SoC (transceiver+FAUST, back and forth) will add 2 samples of delay. Note that the ADAU1787 works at voltage levels up to 1.8V, whereas the Zybo operates at 3.3v. It is therefore necessary to use a level shifter to ensure compatibility between the two boards. The level shifter has to be fast enough to handle the I2S clock (up to 24.576 MHz). We used a LSF0204 level shifter which doesn't induce any significant latency.

	$f_s^{I^2S}$ (kHz)	Ck^{ADC} (kHz)	Latency (μs)	Latency (sample)
SSM2603				
FAUST	48	48	848	41
FAUST	96	96	191	18
ADAU1777				
Bypass DSP	-	768	9.2	-
Bypass I2S	192	768	69.2	13
FAUST	48	96	298.2	14
FAUST	96	96	179.6	17
FAUST	48	768	255.2	12
FAUST	192	768	79.5	15
ADAU1787				
Bypass DSP	-	768	5.9	-
Bypass I2S	192	768	30.3	6
Bypass I2S	768	768	8.4	6
FAUST	48	48	206.2	10
FAUST	96	96	97.2	9
FAUST	48	768	145.6	7
FAUST	192	768	40.8	8
FAUST	384	768	25.4	10
FAUST*	768	768	11.1	8

Table 1. Latency report for different audio codecs. All measurements were made on 24 bits samples, except for the last one (indicated by *) which was made on 16 bits samples. Latency expressed as samples is computed based on the $f_s^{I^2S}$ sampling rate and rounded to the nearest integer.

	LUT (% use)	1 sample comp. time (cycles)	1 sample comp. time (µs)
BRAM synth	66 %	458	3.66 µs
MEM synth	73 %	1477	11.81 µs
BRAM guit. demo	77 %	1922	15.58 µs
MEM guit. demo	84 %	3956	31.64 µs
BRAM lms	64 %	846	7.22 µs
MEM lms	191 %	784	6.27 µs
BRAM tictac	92 %	500	4.00 µs
MEM tictac	95 %	1564	12.51 µs
BRAM comb	34 %	77	0.61 µs
MEM comb	47 %	738	5.90 µs
BRAM violin	109 %	741	6.00 µs
MEM violin	110 %	1955	15.64 µs

Table 2. Memory storage vs. Block Ram storage: comparing performances (Look Up Tables: LUT usage and computation time expressed in cycle of FPGA clock at 125 MHz) for various FAUST programs. Red numbers indicate that the design does not fit on Zybo Z7-10 FPGA. One sample computation time must be less than 20.83 µs at 48kHz (i.e., one sample time).

5.2 Memory Accesses Issues

As mentioned in §3.2, the use of external memory access is often needed for audio programs because FPGA Block Rams do not offer sufficient space for storing delay lines. However, from our observations, a memory access is at least 50 times slower than a Block Ram access. Table 2 shows the impact on latency (i.e., computation time for one sample) of a systematic memory use compared to a systematic use of Block Rams on a set of simple FAUST programs taken from the Amstramgrame website. [22]

These experiments show that using memory during sample computations can possibly increase the computation time of each sample by a factor of 10 (see the `comb` example on Table 2). As explained before, the FAUST compiler has integrated this optimization in the code generated with the `-os2` option. However, in that case, DDR memory should also be used to initialize constants and perform control rate computations (see §3).

5.3 Impact of Initialization

Fig. 7 shows the impact on the FAUST IP size of reporting constant initialization on the ARM processor using `bellN.dsp` as a reference (see Fig 6). Fig. 7 instantiates `N` for integer values between 1 and 30 and compares resources usage (lookup tables and DSPs [23]) for scenarios where initialization is carried on the ARM processor or on the FPGA. It can be seen that no more than 3 resonant filters can be used if FPGA initialization is used, while up to 25 resonant filters can be used if initialization is carried out on the ARM.

These results show that the current version of the `-os2` mode of the FAUST compiler provides adequate optimization of memory accesses for basic FPGA designs. Future work will improve parallelization to reach even better performances.

[22] `https://www.amstramgrame.fr/gramophone/programs/`

[23] DSP here refers to the multipliers integrated in the FPGA.

```
import("stdfaust.lib");

t60 = 30;
pulse = button("gate") : ba.impulsify;

process = pulse : pm.frenchBellModel(N,0,t60,1,2.5);
```

Figure 6. The `bellN.dsp` program used to produce the results presented of Fig. 7. `N` should be replaced by an integer value indicating the number of resonant filters used to provide the bell sound.

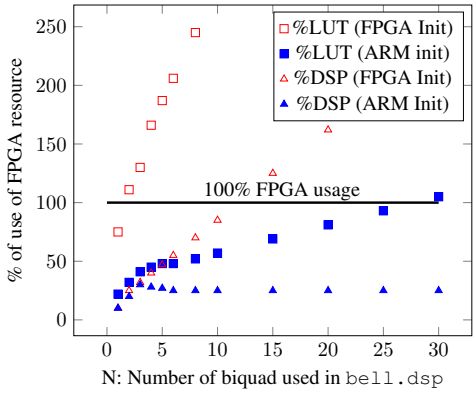


Figure 7. Percentage of Zybo FPGA Look Up Tables (LUTs) and DSPs[23] usage for various sizes (i.e., number of biquads) of the `bell.dsp` FAUST program (see Fig 6), with initialization on the ARM processor (in blue) or on the FPGA (in red).

6. CURRENT AND FUTURE PERSPECTIVES

We are currently working at improving the control of parallelism on the FPGA. The design decisions made for the *one sample* mode have a significant impact because they forbid the pipelining of successive sample computations. Pipelining could possibly solve some performance limitations. On the other hand, it would increase latency, which would be counter-productive. Pipelining successive samples in the FAUST IP would imply to re-think the FAUST IP interface, it could for instance pre-fetch data from memory and overlap more efficiently memory accesses and computations. Along the same lines, we are looking at generating fixed-point code directly from the FAUST compiler which should allow us to generate much more optimized IPs for audio DSP.

The number of audio input/output – 2×2 for our compilation chain – can be increased, but this implies connecting additional codecs and manually adapting the compilation chain (this task remains fairly simple though). Relatively cheap I2S TDM [24] audio codecs ($< \$3$) can now be found on the market. TDM allows us to potentially connect up to eight codecs on the same FPGA GPIO (2 GPIOs if an audio input is needed) opening the possibility to implement systems with an unparalleled number of audio inputs and outputs with potential applications to spatial audio, etc.

Controllers are currently communicating with the Faust IP through the ARM CPU. This implies that the latency of controllers is not necessarily as good as the latency of

[24] *Time Division Multiplexing*

audio streams. This could be improved by connecting the controllers directly to the FPGA.

Finally, the power of FPGAs should allow us to completely get rid of the codec and implement a $\Sigma\Delta$ ADC/DAC directly on the FPGA with simple minimal analog filtering. This would allow us to use much higher audio sampling rate than the ones currently used in our system, opening the door to a wide range of research avenues.

7. CONCLUSION

A large portion of the possibilities offered by FPGAs for real-time audio DSP have yet to be explored. We hope that the work presented in this paper will contribute to making this type of platform more accessible to the audio and music technology research communities. We foresee potential applications for active control (afforded by ultra-low latency) of room acoustics and of musical instruments, spatial audio (with systems potentially offering unparalleled number of audio inputs and outputs), and new approaches to audio DSP with audio sampling rates above 20MHz.

Acknowledgments

This work has been carried out in the context of the FAST ANR project [25] (ANR-20-CE38-0001) funded by the French ANR (Agence National de la Recherche).

8. REFERENCES

[1] Y. Orlarey, S. Letz, and D. Fober, *New Computational Paradigms for Computer Music*. Paris, France: Delatour, 2009, ch. "Faust: an Efficient Functional Approach to DSP Programming".

[2] K. Byun, Y.-S. Kwon, S. Park, and N.-W. Eum, "Digital Audio Effect System-on-a-Chip Based on Embedded DSP Core," *ETRI Journal*, vol. 31, no. 6, pp. 732–740, Dec. 2009.

[3] M. Pfaff, D. Malzner, J. Seifert, J. Traxler, H. Weber, and G. Wiendl, "Implementing digital audio effects using a hardware/software co-design approach," in *10th International Conference on Digital Audio Effects*, 2007, pp. 1–8.

[4] D. Theodoropoulos, G. Kuzmanov, and G. Gaydadjiev, "Multi-Core Platforms for Beamforming and Wave Field Synthesis," *IEEE Transactions on Multimedia*, vol. 13, no. 2, pp. 235–245, Apr. 2011.

[5] J. Zhang, G. Ning, and S. Zhang, "Design of audio signal processing and display system based on SoC," in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*. Harbin, China: IEEE, Dec. 2015, pp. 824–828.

[6] C. Dragoi, C. Anghel, C. Stanciu, and C. Paleologu, "Efficient FPGA Implementation of Classic Audio Effects," in *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. Pitesti, Romania: IEEE, Jul. 2021, pp. 1–6.

[7] K. Vaca, M. M. Jefferies, and X. Yang, "An Open Audio Processing Platform with Zync FPGA," in *2019 IEEE International Symposium on Measurement and Control in Robotics (ISMCR)*. Houston, TX, USA: IEEE, Sep. 2019, pp. D1–2–1–D1–2–6.

[8] M. Verstraelen, J. Kuper, and G. J. Smit, "Declaratively Programmable Ultra Low-Latency Audio Effects Processing on FPGA." in *DAFx*, 2014, pp. 263–270.

[9] E. Pongratz and R. C. John, "Performance Evaluation of MathWorks HDL Coder as a Vendor Independent DFE Generation," 2019, master PhD.

[10] T. Vannoy, T. Davis, C. Dack, D. Sobrero, and R. Snider, "An open audio processing platform using soc fpgas and model-based development," in *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.

[11] T. C. Vannoy, "Enabling rapid prototyping of audio signal processing systems using system-on-chip field programmable gate arrays," Master PhD, 2020.

[12] A. Singhani and A. Morrow, "Real-Time Spatial 3D Audio Synthesis on FPGAs for Blind Sailing," in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. Seaside CA USA: ACM, Feb. 2020, pp. 104–110.

[13] Y. E. Esen and I. San, "Low-Latency SoC Design with High-Level Accelerators Specific to Sound Effects," *International Journal of Advances in Engineering and Pure Sciences*, vol. 33, pp. 78 – 87, dec 2021.

[14] L. Merah, P. Lorenz, A. Ali-Pacha, and N. Hadj-Said, "A Guide on Using Xilinx System Generator to Design and Implement Real-Time Audio Effects on FPGA," *International Journal of Future Computer and Communication*, pp. 38–44, Sep. 2021.

[15] T. Risset, R. Michon, Y. Orlarey, S. Letz, G. Müller, and A. Gbadamosi, "faust2fpga for ultra-low audio latency: Preliminary work in the syfala project," in *Proceedings of the International Faust Conference (IFC-20)*, Paris (France), 2020.

[16] Y. Wang, "Low latency audio processing," Ph.D. dissertation, Queen Mary University of London, 2018.

[17] V. Lazzarini, S. Yi, J. Heintz, Ø. Brandtsegg, I. McCurdy *et al.*, *Csound: a sound and music computing system*. Springer, 2016.

[18] M. Puckette, "Pure data: Another integrated computer music environment," *Proceedings of the Second Intercollege Computer Music Concerts*, pp. 37–41, 1996.

[25] `https://fast.grame.fr`

PRESERVING, RESTORING, AND PASSING DOWN VIDEO-GAME MUSIC FROM THE PAST: THE CASE OF DIRECTMUSIC

Luca A. Ludovico, Alberto Mattea
Laboratory of Music Informatics (LIM)
Department of Computer Science
University of Milan
luca.ludovico@unimi.it

Davide A. Mauro
Department of Computer and
Information Technology
Marshall University
maurod@marshall.edu

ABSTRACT

Video-game music is a form of art that is gaining increasing interest, not only in the technological field but also in musicological research and in the area of Digital Humanities. Characterized by vanishing technological support and by the discontinuing of the software and hardware once used for its reproduction, the establishment of archives and standards that allow the fruition of such artifacts become of crucial importance. The goal of this project is to preserve, restore and pass down video-game music currently available only in an obsolete format. Our main case study will revolve around Microsoft *DirectMusic* technology, adopted in the 90s for composing music for video games and later discontinued. This work can be seen as a first effort at a larger goal that includes a discussion related to the philological process of *what* needs to be preserved and *how*, and the creation of accessible archives.

1. INTRODUCTION

Unfortunately, unlike other forms of musical expression, video-game music is often linked to digital products whose life cycle comes to an end. Not only are commercial products such as video games withdrawn from the market, but the technologies necessary for their execution (hardware architectures, operating systems, etc.) soon become obsolete, preventing the experience of the heritage of musical pieces from the past.

In this sense, the case of *DirectMusic* is particularly relevant. First released by Microsoft in 1996, it became a component of the Microsoft DirectX API. Its goal was to support music and sound effects and provide flexible interactive control over the way they are played. In 1999, *DirectMusic* was introduced as part of version 6.1 of the DirectX library and included in Microsoft Windows operating systems. The first operating system embedding *DirectMusic* was Windows 98 Second Edition. A few years later, *DirectMusic* was deprecated (e.g., it was not available to Windows Vista 64-bit applications), thus making many video-game soundtracks unavailable to new users.

Copyright: © 2022 Luca A. Ludovico, Alberto Mattea et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

In our opinion, the artistic quality, technical characteristics, and historical considerations make some musical artworks worthy of being preserved. The goal of the project is to preserve, restore and pass down video-game music currently available only in an obsolete format. Our main case study will revolve around the *DirectMusic* technology by Microsoft. Adopted in the 90s and used for composing music for video games, it was later discontinued, making it impossible to access such artworks.

This paper is structured as follows: in Section 2 we will address the importance of preserving video-game music, in Section 3 we will analyze the specifications of the *DirectMusic* file format, in Section 4 we will provide details about the design and implementation of a suitable software tool to convert *DirectMusic* soundtracks into currently in-use standard formats (e.g., MIDI), and, finally, in Section 5 we will draw conclusions and list some directions for future work.

2. PRESERVING VIDEO-GAME MUSIC

The first question to answer is *what* do we want to preserve of the original composition.

Dealing with classical music, for example, would be quite straightforward: usually, there is a reference score, namely a list of music symbols, that provides a logical representation of any expected performance. Needless to say, the audio rendering of a given score can greatly vary from performance to performance, depending on the interpretation and the technical skills of musicians, nevertheless, the result is somehow predictable. Even in this well-established field, debates among experts may emerge, e.g., when a piece presents multiple versions, or performance practices diverge from written music. Other genres and composition techniques, for example, jazz or aleatoric music, are less prone to an unambiguous interpretation of a reference score. In these cases, preserving the original composition, or even choosing a paradigmatic performance, can become a hard task.

Non-linear media and dynamic environments such as those of video games clearly emphasize this problem [1]. When the music is rendered in real-time thru the algorithmic manipulation of some building blocks, completely different renderings of the same soundtrack are possible, often depending on the in-game performances of the player. A large number of different music and sound elements can coexist within a unique mixture, and the piece du-

ration itself becomes an unknown parameter. For example, sound samples can be triggered unaltered at specific points in time (e.g., "Game Over" sound) while a non-deterministic engine combines and manipulates small excerpts or whole music sections, originating from symbolic formats, computer-driven performances, or sound samples.

Going back to the example of traditional music, there are basically two ways to identify music information: i) as it regards symbolic content, gaining access to the score; ii) as it concerns audio content, referring to available recordings. Please note that score availability would allow not only the preservation of symbolic content, but also the production of new audio content; and, on the other side, the availability of audio performances in absence of the original score would give the possibility to reconstruct symbolic information, at least to some extent. An example is provided by the transcriptions of jazz improvisations.

Once again, the scenario of video-game music is far more complex. In fact, in most cases, there is no fixed score and no reference performance. While a rendering can be generated and captured during the game-play phase (assuming it is still technically possible to run such a game), this might not be enough to capture the richness of the original composition. Yet, for many video games, gaining access to the source code and the engine used for their creation is impossible, so recordings remain the only viable solution when the entire artifact cannot be preserved.

Now, after *what* should be taken into consideration, a second research question is emerging: *how* can we preserve it?

The subject of digital preservation, intended here in general terms, has been addressed in a great number of scientific works. For example, in a paper dating back to 2001, Chen discussed the so-called paradox of digital preservation: digital information is expected to be maintained intact, but it is accessed in a dynamic use context and, unfortunately, it is plagued by short media life, obsolete hardware and software, slow read times of old media, and defunct documentation and technologies [2]. In 2006, Gladney enunciates the principles for digital preservation, defined as "the mitigation of the deleterious effects of technology obsolescence, media degradation, and fading human memory", addressing prominent epistemological issues, communication-related problems, and big challenges (e.g., persuading information providers to write metadata) [3]. Another relevant work in this field was authored in 2018 by Owens, who presented the theory and craft of digital preservation starting from the traditions and considering the nature of digital objects and media [4].

As mentioned before, providing a comprehensive review of the scientific literature would be out of scope. In the rest of the paper, we will mainly focus on the format chosen to preserve video-game music. The characteristics that such a format should be present are: being standard and currently in use, being well documented and possibly open, presenting no patent or royalties, and having a long-time future perspective. In addition, we will also address the goal of passing down preserved music works by implementing a dedicated web platform.

The choice of the most suitable format can be also driven by the characteristics of the domain to be described. In our case, the context is video-game music whose score is not available and, in general, not uniquely defined, and whose audio rendering may be greatly influenced by the hardware in use and the player's on-the-fly performance.

A possible approach is analyzing the problem from multiple points of view and trying to preserve as much information as possible. In this sense, formats aiming at a comprehensive description of music, such as IEEE 1599, MEI, and MusicXML/MNX, can be employed [5]. Focusing on IEEE 1599, this XML-based standard in its first version was characterized by a 6-layer structure able to accommodate general, logic, structural, notational, performance, and audio information. In the context of this work, this implies the possibility to encode metadata (general layer), symbolic aspects (logic layer), notated music (notational layer), computer-driven performances (performance layer), audio excerpts (audio layer), and even relationships between music objects (structural layer) within a single document. Moreover, IEEE 1599 can be profitably used as an interchange format, too [6].

Another approach when the original asset is not present at all or not available in a standard and in-use format consists in designing an ad-hoc process to acquire as much information as possible and encode it in a current, easily accessible representation. In this paper, we will follow the latter approach and employ the Standard MIDI Files (SMF) format to this goal. The aforementioned challenges are compounded with issues related to copyrights. It is not always clear, looking at software from ten, twenty, or thirty years ago, what happened to the copyright holders and how to acquire those rights.

On one hand, we have game developers and publishers who use licensed music in their products, and, on the other, we have composers and sound designers that are hired for specific titles, and those compositions exist only for the video game.

While it is true that in recent times more and more videogames soundtracks are released independently from the original game this does not cover the entire industry.

3. THE DIRECTMUSIC FORMAT

3.1 History

In 1995 Microsoft introduced DirectX, a set of application programming interfaces (APIs) with the aim of handling tasks related to multimedia, especially game programming and video. DirectX allowed developers of games and other interactive content to access specialized hardware features without having to write hardware-specific code [7]. Among this collection of APIs, the *DirectSound* one was intended to provide a low-latency interface to sound card drivers. Since it was essentially a low-level interface, it soon became clear that a higher level of abstraction was needed to simplify and standardize the process of sound and music creation.

To this end, in 1996 the *DirectMusic* API was introduced, initially released as an ActiveX control called Interactive

Music Architecture (IMA) [8]. *DirectSound* focused on the capture and playback of digital sound samples, while *DirectMusic* managed message-based musical data. An online article describing the new possibilities of the format can be found in [9].

During its lifetime, such a format was used by many composers and programmers. A non-comprehensive list of video games, which includes major publishers, can be found in a dedicated discussion forum dating back to 2002 [10].

Nevertheless, the lifetime of the *DirectMusic* API was relatively short. In 2006, only 10 years after its official release, it became deprecated. Consequently, not only *DirectMusic*-compatible video-game music was no more produced, but the already existing compositions utilizing such a technology quickly met a fate of decline and abandonment. The goal of this work is to preserve, revive, and pass down this heritage.

3.2 Data Structures

The file encoding is based on the Resource Interchange File Format (RIFF) bitstream format [11], employing data structures known as *chunks* for storing data. A *chunk* is a fragment of information that contains a header carrying some parameters (e.g., the type of chunk, its size, etc.) followed by a variable area containing the data payload. The structure of a RIFF file is very simple: it can be represented as a tree, where each element (*chunk*) has an id, a dimension, possibly a type, and a payload. Unlike a normal tree, where each node has a list of pointers to its subnodes and subnodes reside in a separate area, in RIFF each *chunk* contains the *subchunks* in its payload. The file is therefore presented as a single root (or top-level) chunk that contains all the others, and the header of this *chunk* is also the header of the file. Many multimedia file formats, such as PNG, IFF, MP3, WAV, and AVI, are *chunk*-based.

In *DirectMusic*, the main structures contained in the RIFF file are known as *Forms*, whose most common types are:

- *Segment*, the container for an entire musical excerpt;
- *Style*, carrying the information on patterns that can be dynamically combined at run time;
- *Track*, a list of music events or data to control the performance;
- *Band*, used to define an instrument;
- *Reference List*, a mechanism to connect either different files or *Forms* of a file.

A *Segment* represents an entire piece of music. From a technical point of view, it is a chunk with its id set to RIFF. A *Segment* is usually stored in a separate file, with the extension .sgt (for playback only) or .sgp (for editing with *DirectMusic Producer*). Within the segment, there is a header, which indicates how the song should be played (start and end timestamps, loops, etc.), and the payload is a list of one or more *Tracks*.

The concept of track in *DirectMusic* is more generic than in MIDI: there are many types, and, despite being grouped

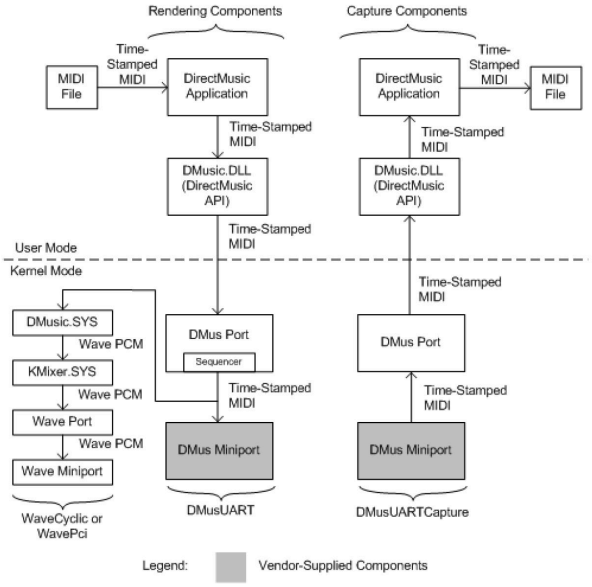


Figure 1. The structure of the *DirectMusic* API. [12].

under a common umbrella, each type has different behavior. Thus, for a parser, it is fundamental to check the *fccType* field in the header. Among the most important *Track* types, it is worth mentioning objects that are simple lists of events (*seqt*), those that give information on tempo changes (*tetr*), those that execute commands that can alter the reproduction (*cmnd*), and those that contain reusable styles and patterns (*sttr*). All *Tracks* are read simultaneously during playback, and events, commands, tempo changes, etc. have an absolute timestamp that defines when they should occur.

One of *DirectMusic*'s strengths is native support for dynamic music, a composition generated on the fly based on the current situation in the game. This result is achieved with the use of *patterns*, namely short musical sections that are automatically selected, combined, and transposed on the chords chosen by the composer. The fundamental parameter that controls such a process is called *groove*, i.e. a number that should give a measure of the intensity of the situation: for example, a high value of groove is typical in a battle or in the fight against the final boss. The groove level can also be varied within the song itself, with a command given by a *cmnd* track: in this case, the aim is simply to build variations of music themes automatically, so as to ease the composer's work.

3.3 Technical Remarks

The structure of the *DirectMusic* API is shown in Figure 1

As it concerns sound synthesis, music can be performed either in hardware, using the Microsoft GS Wavetable SW Synth, or in a custom synthesizer. Thus, audio tracks are rendered by a virtual synthesizer that allows for unlimited channels and rich control messages, expanding the concept of discrete Control Change available in MIDI.

Concerning virtual musical instrument programs, *DirectMusic* adopts DLS [13], a set of standardized file formats

	DMP	*sgt2wav*	*Ficedula*	*libdmusic*
Licence	proprietary	free	unknown	free
Runtime	DirectX	DirectX	DirectX	independent
Operating System	Windows 32 bit	Windows	Windows	Windows/Unix
DM Support	complete (play and modify)	complete (only play)	complete (only play)	subset of DirectMusic 8
Status	abandoned	abandoned	abandoned	under development
Retrieval	waveforms and MIDI subset of sound events	waveforms	waveforms	waveforms

Table 1. Comparison between software solutions that support *DirectMusic*.

for digital musical instrument sound banks developed first by the Interactive Audio Special Interest Group (IASIG), and then by the MIDI Manufacturers Association (MMA). DLS is a versatile format, which in addition to samples, can also define synthesized instruments and supports transformations such as envelopes. The sound font is connected to the *Segment* through a *Form* of type *Reference List*; the single instrument is instead identified through a set of *Band Track* and *Band*. Although a documented standard, the DLS format is no longer widespread today, and software support is also limited, especially on Linux. For this reason, in a context of preservation, it is preferable to convert DLS sound fonts into the universally supported SF2 format.

4. THE DIRECTMUSIC CONVERTER

In this section, we will first review some already existing solutions to convert *DirectMusic* music pieces, and then we will describe our proposal.

4.1 Background

There are a number of solutions currently available to render and export original *DirectMusic* files, but they all have specific drawbacks or limitations. A first distinction concerns the methods that rely on Microsoft's runtime and those that don't. Almost all software that uses *DirectMusic* access it through the Software Development Kit and the runtime provided by Microsoft within DirectX. This means that their functioning is inextricably linked to Microsoft's choice to maintain support and backward compatibility. Furthermore, this aspect precludes the use of *DirectMusic* on operating systems other than Windows. This category embraces, in addition to game engines, plugins that allow you to listen to *DirectMusic* pieces through general-purpose media players (an example is a plugin for Winamp developed by user *Ficedula* from the Final Fantasy community), dedicated reproducers/converters such as *sgt2wav*, and *DirectMusic Producer*.

DirectMusic Producer (DMP) is the original software provided by Microsoft for editing. Last updated in 2002, nowadays even finding a working copy of the software can become a challenge, and it will most likely require a legacy installation of Windows. *DMP* allows exporting a rendering of the performance in a WAV file at the cost of the original "structure" of the document. On the contrary, it is possible to capture MIDI events during a performance.

Another software is *sgt2wav*, a command-line application for Windows. Written in C++, it just requires the DirectX runtime and an audio device. A rendering of the performance is exported with the same limitations as *DMP*. This software will also cease to work when Microsoft will discontinue the *DirectMusic* support in DirectX.

There are also a few independent projects that try to reimplement the format support from scratch, covering operations from parsing to rendering. They are unrelated to support policies, a specific operating system, and version, but, in general, they offer a lower level of compatibility. The only significant example of this category is *libdmusic*, an Open Source (MIT License) library that rewrote from scratch the support for the *DirectMusic* format. For this reason, it is limited to a subset of the functionalities available in version 8. Like the other solutions mentioned here, its goal is to provide a rendering of the output.

A comparison between the mentioned approaches is presented in Table 1.

4.2 dmproj

Our proposal for extracting and preserving information from the *DirectMusic* format relies on software developed in Python 3.x, called *dmproj*. The source code is available on Zenodo under GNU General Public License v3.0. [1]

The software has been conceived with a modular architecture:

- The *riffparse* module recursively parses a RIFF file in order to extract the chunks and their (potential) subchunks.
- The *dmparse* module receives in input a chunk and exposes the underlying structure (e.g., a *Segment*);
- The *dm2midi* module, finally, takes care of generating MIDI events from a Segment. It is important to note that there might be the need to generate multiple MIDI messages for a single *DirectMusic* "event" (e.g., control change messages that correspond to a curve described with only one command).

The output of *dmproj* is a Standard MIDI File (SMF) type 1 [14]. There are multiple advantages to adopting such a format. First, even if standardized decades ago, it is still in use, being supported by many media players and often

[1] https://doi.org/10.5281/zenodo.5952340

```
RIFF 42926 DMSG          /* Root RIFF structure, of type segment (.sgt file) */
.segh 24                 /* Segment header */
.guid 16                 /* Global unique identifier, used by the DirectMusic runtime to load the appropriate segments */
.LIST 116 sgdl           /* Segment data list (undocumented) */
..segd 104
.vers 8                  /* File version */
.LIST 148 UNFO           /* Generic metadata (name, comment, etc) for the whole file */
..UNAM 18
..UCMT 110
.LIST 42562 trkl         /* Track list */
..RIFF 236 DMTK          /* Chord track */
...trkh 32               /* Track header (specifies the track type) */
...LIST 172 cord         /* List of chords */
....crdh 4
....crdb 132
....crdt 8
...ctdc 3
..RIFF 72 DMTK           /* Tempo track */
...trkh 32               /* Track header (specifies the track type) */
...tetr 20               /* List of tempo changes and corresponding timestamp */
..RIFF 1110 DMTK         /* Events track */
...trkh 32               /* Track header (specifies the track type) */
...LIST 30 UNFO          /* Generic metadata (name, comment, etc) for the track */
....UNAM 18
...seqt 876              /* Events container */
....evtl 744             /* Note events */
....curl 116             /* Control events */
...LIST 136 psql         /* Proprietary undocumented structure used internally by DMP. Doesn't affect the actual music */
....psqc 48
....cvau 4
....cvsu 8
....cvsu 8
....cvsu 8
....cvsu 8
..[...]                  /* Other events tracks (usually one per voice/channel) */
..RIFF 28300 DMTK        /* Band track (external encapsulation) */
...trkh 32
...RIFF 28248 DMBT       /* Band track (actual) */
....LIST 28236 lbdl      /* Band list */
.....LIST 776 lbnd       /* Band (external encapsulation) */
......bdih 4
......RIFF 752 DMBD      /* Band (actual) */
.......guid 16           /* Global unique identifier */
.......LIST 24 UNFO      /* Generic metadata (name, comment, etc) for the band */
........UNAM 12
.......LIST 684 lbil     /* Instrument list */
........LIST 242 lbin    /* Instrument */
.........bins 40         /* Instrument properties (allowed range, channel, volume, transposition, etc) */
.........LIST 142 DMRF   /* Soundfont reference */
..........refh 20
..........guid 16
..........date 8
..........name 20
..........file 18
..........vers 8
.........jzfr 32
........[...]            /* Other instruments */
.....[...]               /* Other bands */
```

Figure 2. Commented dump of a *DirectMusic* music piece (excerpt).

used also for interchange purposes. In addition, with respect to WAV files, SMFs are much lighter, as they contain commands to instruct a synthesizer in producing audio instead of containing audio samples. This approach can also represent a drawback since the final performance is deeply influenced by the quality of audio components (in particular, the MIDI synth) and the availability of high-quality sound samples. Nevertheless, the representation of a performance in terms of commands instead of a fixed, pre-calculated sequence of samples is closer to the original approach of *DirectMusic*. Finally, MIDI performances are not burdened by performing rights.

Conversion of all *DirectMusic*'s features to MIDI requires going beyond the limits of the single standard MIDI file. For example, a music piece could employ more than 16 channels or include waveforms encapsulated directly or through an external reference. For the former problem, a possible solution is to divide the *DirectMusic* song into several MIDI files based on blocks of 16 channels (or 15, if you want to respect the convention that reserves Channel 10 for percussion). To overcome the latter problem, you can export the waveforms in a sound font and insert the suitable control-change and program-change commands in the event flow. Despite these limitations, much of the original information can be kept, thus offering a much more adequate recovery tool than the alternatives.

4.3 Example

In order to demonstrate the efficacy of the proposed solution, we have converted a .sgp file into a .mid file. Both files are publicly available, [2] together with a sound font to be loaded in the MIDI synth for reconstructing the original sounds.

Figure 2 shows the dump of the original music piece, limited to an excerpt for the sake of brevity. Some comments have been introduced to make data structures clearer to the reader.

In this case, the composition is based on a *Segment* only, thus the conversion can be complete.

The difference in size between the original (41.9 KB) and

[2] https://lim.di.unimi.it/media/smc2022_examples/cityfast.zip

the destination file (6.7 KB) shows that the .sgp format was not very efficient at encoding information.

5. CONCLUSIONS AND FUTURE WORKS

In this contribution, we approached the problem of the preservation of video-game music both from a technical and technological point of view and from a philosophical one. Answering the questions of *what and how* we are going to preserve this cultural heritage informed us on the technical decisions that have to be made in the implementation stage.

We looked at the case study of *DirectMusic*, a format that had certain popularity but is now extinct. We presented its characteristics and developed software that allows for a conversion from this format to a currently supported one (MIDI).

The natural development of this project consists in supporting all *DirectMusic* features. In this sense, the main directions are:

1. The extension to the next version of the format, the one used in conjunction with DirectX 8 and 9. Version 9 does not introduce really significant changes, so the two versions can be considered essentially the same. The main changes concern the addition of new fields at the bottom of some data structures. It would therefore be necessary to compare the headers supplied together with the related SDKs to understand the differences and expand the related classes within the decoder;

2. The full support of styles and patterns, which are currently implemented as stubs. A known limitation of the software is the ability to convert only sequence-based music pieces, which can be (and actually are) transformed into MIDI without loss of information. In order to extend its potential, the software should implement the random choice of the pattern to be played, the setting of the groove level, and the transposition through chords. In this way, the piece would be converted in only one of its possible forms since MIDI does not manage the variations. A solution could be to generate all the possibilities;

3. The management of more atypical types of curves used in control messages, such as sinusoidal ones. Even if little used in real games, they are still part of the format and allow interesting creative possibilities.

After improving the conversion tool, the next step will be to organize an archive that allows these heterogeneous materials to be stored, searched, and experienced by users.

While the software presented here is just a first step in a bigger effort, we hope it will elicit a conversation around the topic of the preservation of the digital heritage of video games music.

Acknowledgments

This project has been partially supported by the contribution of the Marshall University Faculty Senate Research Committee (FSRC) Funding.

6. REFERENCES

[1] K. Collins, "An introduction to the participatory and non-linear aspects of video games audio," *Essays on sound and vision*, pp. 263–298, 2007.

[2] S.-S. Chen, "The paradox of digital preservation," *Computer*, vol. 34, no. 3, pp. 24–28, 2001.

[3] H. M. Gladney, "Principles for digital preservation," *Communications of the ACM*, vol. 49, no. 2, pp. 111–116, 2006.

[4] T. Owens, *The theory and craft of digital preservation*. Johns Hopkins University Press, 2018.

[5] A. Baratè, G. Haus, and L. A. Ludovico, "State of the art and perspectives in multi-layer formats for music representation," in *Proceedings of the 2019 International Workshop on Multilayer Music Representation and Processing (MMRP 2019)*. IEEE CPS, 2019, pp. 27–34.

[6] A. Baratè, L. A. Ludovico, D. A. Mauro, and F. Simonetta, "On the adoption of standard encoding formats to ensure interoperability of music digital archives: The IEEE 1599 format," in *DLfM '19: 6th International Conference on Digital Libraries for Musicology*. ACM, 2019, pp. 20–24.

[7] Microsoft, "Microsoft ships DirectX 5.0," https://news.microsoft.com/1997/08/04/microsoft-ships-directx-5-0/, 1997, online; accessed 23 January 2022.

[8] T. Buttram, "DirectX 9 audio exposed: Interactive audio development, chap. beyond games: Bringing DirectMusic into the living room," 2003.

[9] T. Hays, "DirectMusic for the masses," *Gamasutra.com*, 1998.

[10] B. Lynne, S. Bottcher, S. Maloney, S. Morgan, J. Kaae, and D. Yackley, "Games that use DirectMusic," https://www.freelists.org/post/directmusic/Games-that-use-DirectMusic, 2002, online; accessed 23 January 2022.

[11] IBM Corporation and Microsoft Corporation, "Resource Interchange File Format," in *Multimedia Programming Interface and Data Specifications 1.0*, 1991, http://www.tactilemedia.com/info/MCI_Control_Info.html, online; accessed 23 January 2022.

[12] Microsoft, "DirectMusic API Documentation," https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/midi-and-directmusic-components, online; accessed 30 March 2022.

[13] S. J. Welburn and M. D. Plumbley, "Rendering audio using expressive MIDI," in *Audio Engineering Society Convention 127*. Audio Engineering Society, 2009.

[14] MMA, *The Complete MIDI 1.0 Detailed Specification*. Los Angeles, CA: MIDI Manufacturers Association (MMA), 1996.

DESIGNING SOUND REPRESENTATIONS FOR MUSICOLOGY

Pierre Couprie
RASM-CHCSC, Paris-Saclay University
pierre.couprie@univ-evry.fr

ABSTRACT

This article describes the use of representations based on data extracted from the audio signal (sonogram, chromagram, audio descriptor) in musicology. A first part gives the historical context of these representations and discusses their transfer from the field of exact sciences to musicology. In the second part, we propose three new types of representation particularly effective in musical analysis: visualizations based on waveform and amplitude, a multilayered sonogram and chromagram, and an interface to assist the interpretation of self-similarity matrices for the analysis of musical forms and microstructures.

1. INTRODUCTION

There are many forms of representation of the sound signal: from the waveform to higher-level representations such as FFT visualization, audio descriptors or the chromagram. Some of them are commonly used by musicians or musicologists, others are used more rarely. Some representations are easily applicable in musicology, others are more adapted to musical acoustics and their interpretation is often difficult for a researcher in human sciences[1]. For the last fifteen years, we have been working on the improvement of these visualization techniques and on the development of representations dedicated to musicology. The aim of this article is to present some representations and layouts particularly effective for musicological research and more specifically for musical analysis. These representations have been produced using the iAnalyse version 5 software[2].

In the first part of the present paper, we will present the context related to the use of these representations in a musicological framework. In the second part, we will introduce three types of representation allowing actual interaction between the knowledge handled by the musicologist, musical perception and data extracted from the audio signal.

2. CONTEXT

2.1 Signal Representations in Musicology

In musicology, representations of the audio signal and the information it contains are often used in the practice of musical analysis. This practice, which is largely empirical [2], is based on the use of a visual support enabling the formalization of musical data in order to study them. However, a significant part of the musical repertoire has no visual support - such as a transcription or a score - or an incomplete support - as in the case of music mixing acoustics and technologies[3], for example electroacoustics [3] or popular music. Creating this visual medium has therefore become indispensable. The study of complex sound materials, often used in experimental creation, presents problems of memorization. In the case of long works, musical analysis can be particularly fastidious and the use of sonograms or self-similarity matrices allows the analyst to overcome this problem. In the same way, a visual support greatly facilitates the discovery of a work by establishing a hierarchy for the listening process or by facilitating identification of significant events, moments of stability or singularities. The use of detailed representations also enables a more specific study of microevolutions of the material[4] such as inflections and variations of the musical interpretation of a melodic line, differences in pitch, modifications of timbre by effects of granularity, variations between channels reflecting effects of space or phase shifting etc. Finally, one of the major difficulties for the musicologist working on audio recordings resides in the communication of the work. If the use of score excerpts is a good way to illustrate a demonstration, in the case of music without score or the analysis of non-notated musical characteristics, visualization has become an indispensable support.

2.2 Three Types of Audio Signal Data Used in Musicology

The objective is not to provide an exhaustive state of the art but to present the most important research activities in the field of musicology[5].

2.2.1 Waveform and Amplitude

As early as the 1960s, Pierre Schaeffer used a representation of the evolution of the sound amplitude produced using a bathygraph [5]. This graph allowed him to observe the permanence of the intensity curve of certain sounds even after having removed the attack.

[1] Some books are even exclusively designed for musicologists with a solid background in exact sciences, for example the Handbook of Systematic Musicology [1].
[2] http://ianalyse5.pierrecouprie.fr.
[3] This trend has increased over the last 30 years with the development of studies on repertoires that were previously rarely studied, such as popular or traditional music.
[4] For a thoroughly detailed use of signal visualization techniques for musicology, see Michèle Castellango's book [4].
[5] There are in particular numerous research projects applied to musical acoustics, to the study of musical instruments or to creation. They are not presented in this paper, which focuses on musicological uses.

In the representation of *Rosace 5*, an extract from *Vibrations composées* by François Bayle (1973), produced by Dominique Besson, the waveform is drawn in two superimposed solid curves [6]. Each curve represents the amplitude of a channel colored in red for the right channel and in green for the left channel. The superposition of the two amplitude curves facilitates the analysis of the correlations between audio channels. This representation of the sound amplitude associated with a transcription of the various sounds is similar to the transcription of Karlheinz Stockhausen's *Studie II* (1954) performed by the composer [7].

The waveform - or an equivalent representation derived from the RMS amplitude - is the representation usually used in audio software to visualize intensity and manipulate the sound. It is therefore the first representation of the signal available to the musicologist. It facilitates visualization of the formal elements related to amplitude. Thus, in her analysis of Christine Groult's *La condition captive* (2003), Ana Dall'Ara-Majek uses the waveform as a background for the representation of formal segmentation [8]. In this work, the waveform underlines the different parts through opposition of intensities or morphological variations.

Although the waveform may not provide information on the timbre parameter, it does highlight the balance of intensities, breaks and abrupt variations that are often rhythmic or even formal markers.

2.2.2 Sonogram

Emile Leipp is probably one of the first to adopt spectrum analysis for sound recognition [9]. He presents the sonogram as the ideal tool for sound analysis, pointing out that the acoustic image corresponds exactly to the mental image suggested by perception. His examples are most instructive and convincing, but is this technique applicable in a musicological context? In 1984, Robert Cogan proposed a method of musical analysis based on the study of spectral graphs [10]. From these graphs, the author deduced color registrations of timbre representing a schematized spectral distribution that allows an analysis of the musical form and also an identification of types of sound and their transformation. However, their representation was poor quality and the method of description did not really result in an accurate analysis of spectromorphologies, but rather in broad spectral block opposition. In spite of its limitations, this approach had the merit of raising the question of the use of the sonogram in musical analysis.

After Cogan, Martha Brecht [11] and Mara Helmuth [12] were among the first researchers to propose an analytical method adapted to electroacoustic music and based on a multidimensional representation including the sonogram. Brech uses a representation combining three elements: the sonogram, musical structures on several levels and the amplitude. In addition, the author often superimposes textual comments on the sonogram. Helmuth's device combines five elements stacked from bottom to top: a sonogram, pitches on a staff, a reading of musical phrases above the staff, a graph of amplitude and a line of textual comments. According to the author, this layout provides a clear visualization of the elements essential to the analysis of the work.

In 2010, Michael Clarke proposed the use of an interactive stereophonic sonogram to visualize the frequency balance between channels [13]. This stereophonic sonogram offers an additional level of visualization of sound characteristics for musical analysis.

In 1990, Waters and Ungvary list the potential uses of spectral representations for music [14]: from composition to performance, including analysis and music teaching, the sonogram is presented as a representation especially suited to these musical practices.

However, although the waveform or the amplitude representation prove to be relatively reliable, two limitations must be underlined. The first limitation, as Curtis Roads points out [15] is that the sonogram is only an estimation of the spectrum analysis by the machine and not an image of what our perception analyses. The second concerns the adjustments and coloring of the sonogram which, far from being trivial, make it particularly suitable even when it concerns recordings that are difficult to visualize because of complex sounds or those close to noise, use of extreme amplitudes, continuity of the texture or very slow variations of the material, etc. These are all parameters of musical analysis that are barely visible when using the default settings of the sonogram.

2.2.3 Audio Descriptors

There are three types of descriptor [16]: low-level descriptors (spectral brightness, spectral dispersion, noise level in the signal, etc.), medium-level descriptors offering an estimation of some musical parameters (fundamental frequency, inharmonicity, beats, etc.) and high-level descriptors (recognition of harmony, tonality, emotion, musical genre, etc.). Only the first-level descriptors use robust algorithms. The results obtained with second and third level descriptors often include errors and are adjusted to specific repertoires, so their generalization is not possible in musicology.

As early as 1957, Charles Seeger used a device to extract the fundamental frequency and the intensity per range in order to facilitate the transcription of melodic lines [17]. It is only since the 1990s that audio descriptors have been developed to allow the classification of large digital sound databases. Although their use in musicology is still struggling to become common practice, there are many examples of their application. Thus, Kerstin Neubarth et al. list a group of musicological practices that benefit from the use of audio descriptors, from transcription to musical analysis, including the study of perception of works [18].

In 2009, Tae Hong Park et al. presented a MATLAB toolkit (EASY) for the analysis of electroacoustic music using audio descriptors [19]. For example, they propose the use of 3D timbregram representation based on the association of several descriptors and clustering techniques to analyze the timbre.

2.3 Technology Transfer to Musicology

Even if some technologies are now commonly used in musicology, the transfer of methods from exact sciences to human sciences poses serious challenges.

The first difficulty lies in understanding their methods and the way in which the computer algorithms that implement them function. The musicologist uses the sonogram in order to find answers to questions that emerge during analytical perception. They do not seek to demonstrate the obvious - what any expert listener is able to perceive - but to complete their analytical knowledge, even to question it. The sonogram will therefore generally be used to analyze complex textures, mixtures of timbre that are difficult to identify even for an expert, interfering noises that allow the recording to be located, etc. These will often be non-standard recordings that require specific FFT computation settings and coloring. In addition to understanding the calculation of this FFT, it must also be situated in the musical domain, not only in the domain of sound or acoustics. The transfer of these technologies must therefore be accompanied by a transfer of method. Using a sonogram in musicology does not only mean using technology coming from acoustics and computer science, but also means using a digital method of musical analysis.

The second difficulty is that musicologists are generally limited to using tools developed for other sciences that are basically inadequate when applied to their scientific methods. Extracting an audio descriptor and visualizing it in the form of a graph, as is possible with several software programs, is not enough to enable its scientific use in music. It is necessary to develop visualizations of the descriptor adapted to its musical interpretation. In other words, a simple graph is perfect for understanding the mathematical characteristics of the data, but does not give any clues regarding a possible musical interpretation - for example in terms of morphologies, formal transitions, musical variation, etc.

The third difficulty is related to the previous one. Many research projects conducted in exact sciences would be a precious help in musicology. Unfortunately, the production of software is often limited to a few Python scripts that are difficult to use for a musicologist. On the one hand, the researcher in social sciences and humanities is not trained to use them and, on the other hand, it requires a high level of constant scientific monitoring in fields for which the musicologist does not have the time.

The fourth difficulty concerns the error transmitted by the data. The sonogram is only an estimation of a spectral content that is often far from our musical perception. Similarly, the audio descriptors often generate many errors, even for low level descriptors - for example the spectral centroid values when the sound amplitude decreases. These errors are unavoidable and often seem incompatible with the scientific methods of the musicologist. What may seem an important step forward for a researcher in exact sciences - for example the extraction of data from a vast musical collection, even with a margin of error - is unacceptable for the musicologist who cannot be satisfied with approximations when analyzing a corpus.

The application of these techniques is therefore not to be found in a simple transfer between the exact sciences and musicology but in the development of new musicological methods. This is the role of digital musicology that differs from computational musicology, which is often based on the transfer of methods and technologies. Digital musicology is interdisciplinary, which is not without its problems, since even if the subject of research, music, is also interdisciplinary, its study remains deeply rooted in certain disciplines of the humanities - generally history, analysis, aesthetics, philosophy, and anthropology.

For all of these reasons, we have developed the software iAnalyse which allows the musicologist to use complex technologies with representations adapted to their field of research[6]. We have already discussed the basic functions and possible utilizations in several publications [20]. In the third part, we will focus on some new forms of spectrum representation or descriptors that are particularly effective for music analysis.

3. EXAMPLES OF REPRESENTATION

These three types of representation are the result of several recent research projects in musical analysis and are integrated in the iAnalyse software.

3.1 Amplitude and Augmented Waveform

As we have discussed previously, the representation of the sound amplitude or the waveform is probably the most commonly used visualization in the musical domain. It facilitates identification of the temporal relationship between sound characteristics such as attacks, musical intensity, morphologies or occurrence of silences. In the iAnalyse software, we have experimented with little used or forgotten representations. Figure 1 shows three examples of waveform or sound amplitude visualization. The top representation displays the two channels and their deviation[7] (red curve). The middle representation displays the two channels on different layers, it facilitates the visualization of their differences. Finally, the bottom representation provides an overview of three audio descriptors: the spectral centroid (y-axis) which is a good estimate of the brightness of the sound, the RMS amplitude (height) and the zero-crossing rate (colors) which measures the noise level (and therefore the inharmonicity) of the signal. This last type of representation is based on the brightness standard deviation (BStD) proposed by Mikhail Malt and Emmanuel Jourdan for the study of timbre [22].

[6] Technologies implemented in iAnalyse 5 are derived from previous work on another software program, EAnalysis, developed in partnership with the MTI[2] of De Montfort University [21].

[7] The deviation is computed from the RMS amplitude as for the other graphs in the figure.

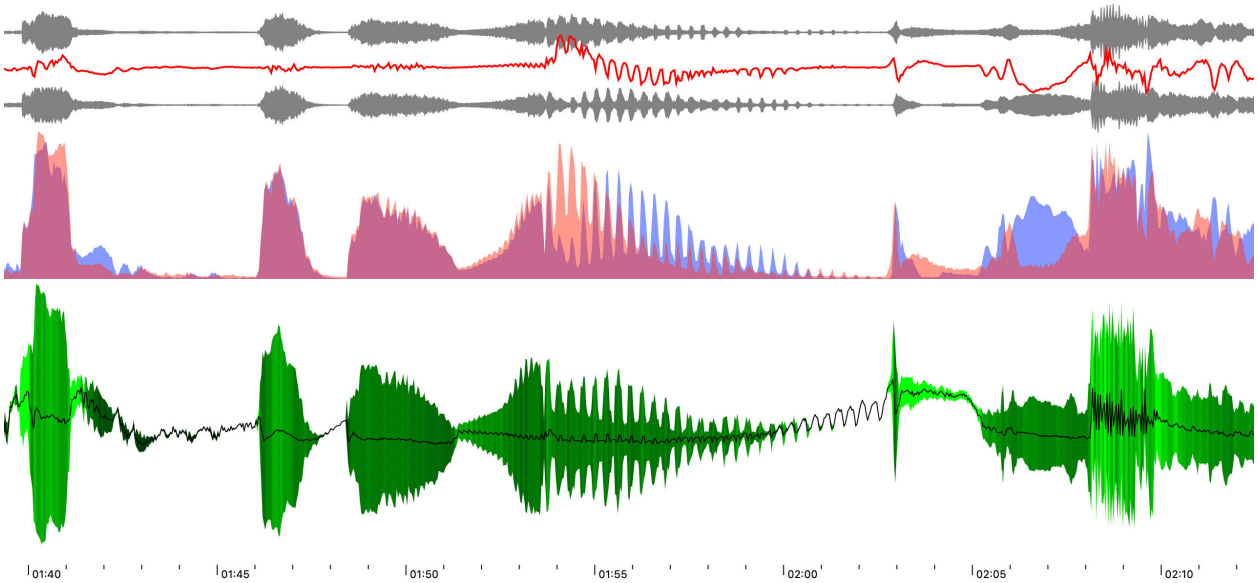


Figure 1. Three examples of amplitude visualization applied to an excerpt from *Ah Dulcinée* by Alain Savouret (from top to bottom): waveform (gray) and channel deviation (red curve), RMS amplitude on two layers, a combination of three audio descriptors (y: spectral centroid; height: RMS amplitude; colors: ZCR)

These three figures demonstrate the impact of the visual element on the interpretation of the data. For example, the central figure presents an immediate view of the balance between the channels and the lower figure illustrates the melodic profile. This immediacy is very useful for the musician or researcher who, at a single glance can identify moments on which they should focus particular attention.

3.2 Multilayered Sonogram and Chromagram

In 2019, the composer and musicologist Lin-Ni Liao initiated a research project[8] on the sheng in contemporary music. This instrument, which has a strong presence on the Asian continent, has been the object of much enthusiasm from contemporary composers for over 20 years. During the summer of 2019, we carried out a free improvisation session with the musician Li Lin Chin and the composer and musician Benjamin Levy at Ircam using the improvisation software OMax. Our goal was to study the behavior of the software with the sheng improvisations. We recorded the audio in multitrack and then developed representations to visualize interactions between the musician and the OMax software. Two visualizations have proved to be particularly relevant: the multilayered sonogram (figure 2) and the multilayered chromagram (figure 3).

The first figure simply consists of a superposition of transparent logarithmic sonograms. Each sonogram is tinted with a color enabling a comparison of the musician's playing (in red) and the three iterations of OMax[9] (here only two iterations are active in green and blue). The processes used by the improvisation software are thus immediately recognizable: transposition, recording-playback with a delay, densification by duplication and their combinations.

The second figure (figure 3) uses the same principle of superposition from a chromagram[10]. In this case, as the sheng is principally a harmonic instrument[11], this type of representation proves to be most suitable for recording. In order to improve readability, we decided to draw the dynamic profile (amplitude RMS) directly on the pitches, which permits correlation of data between the different tracks (red, green, blue and orange). Here also, the interaction between the musician (in red) and the software (the other colors) is clearly visible whereas the spectral texture is somewhat complex: response of the musician on some morphologies, morphological differentiation of pitches, improvisation strategies between the musician and each iteration of the software, software fragmentation of sounds, etc.

These two representations therefore provide a tool for music analysis using visualizations, known to acousticians but adapted to musicology.

[8] http://www.tpmc-paris.com/sheng-research-2019-2020/.

[9] OMax is an automatic improvisation software program developed at Ircam. It has generated many variations such as SoMax or, more recently, DyCI2 (see: https://www.ircam.fr/projects/pages/omax/).

[10] The chromagram represents the intensity of each note computed from the FFT and plotted for each octave.

[11] Some blowing effects and key noises were also used by the musician, but they are not relevant in the excerpt in Figure 3.

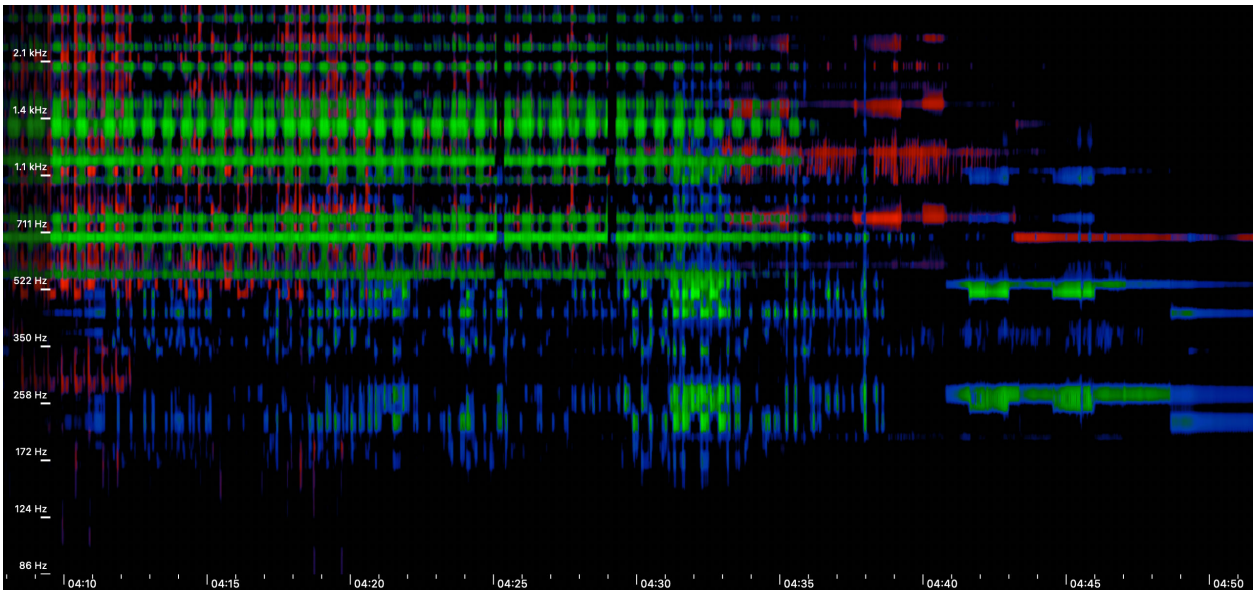


Figure 2. Multilayered sonogram to visualize correlations between recorded audio tracks.

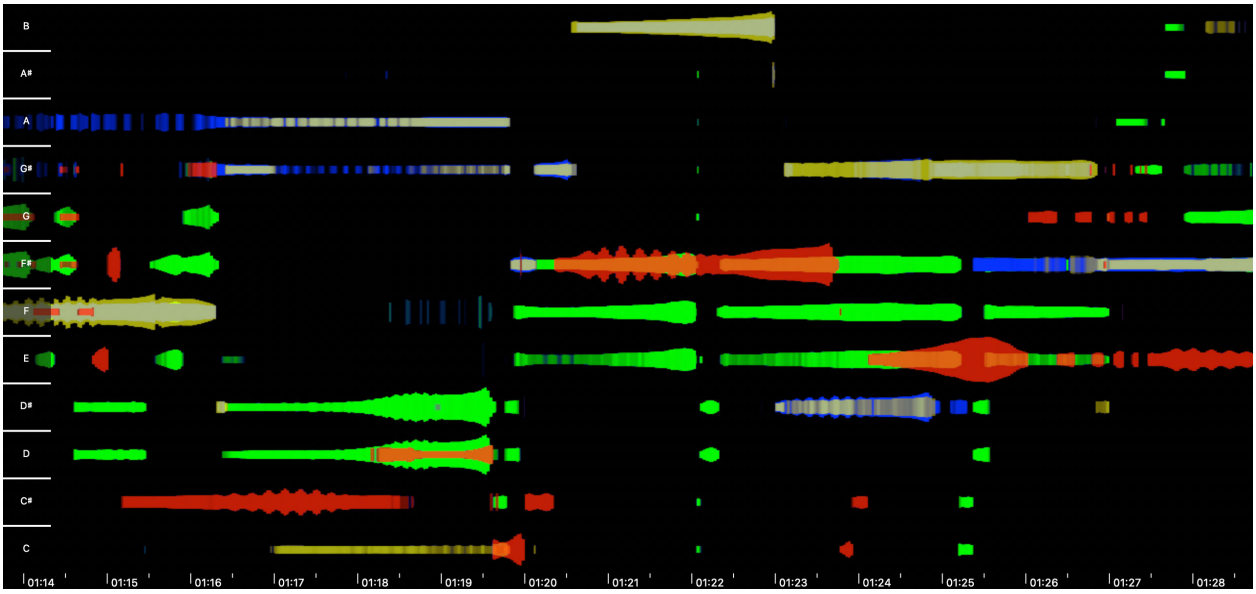


Figure 3. Multilayered chromagram to visualize correlations between recorded audio tracks.

3.3 Self-Similarity Matrix and Distance Estimation

Self-similarity matrices are computed from the FFT data or from a set of audio descriptors. The calculation is performed in three steps (figure 4). The first step (A) consists in creating a matrix (table) having the same values on the x-axis and y-axis and calculating distances between the values. The second step (B) consists in mapping the numerical values against the color gradient values. Finally, the last step (C) consists in coloring the matrix in false colors in order to enhance interpretation. The abscissa and ordinate values being the same, the matrix is symmetrical in its diagonal and the two triangles are therefore symmetrically identical.

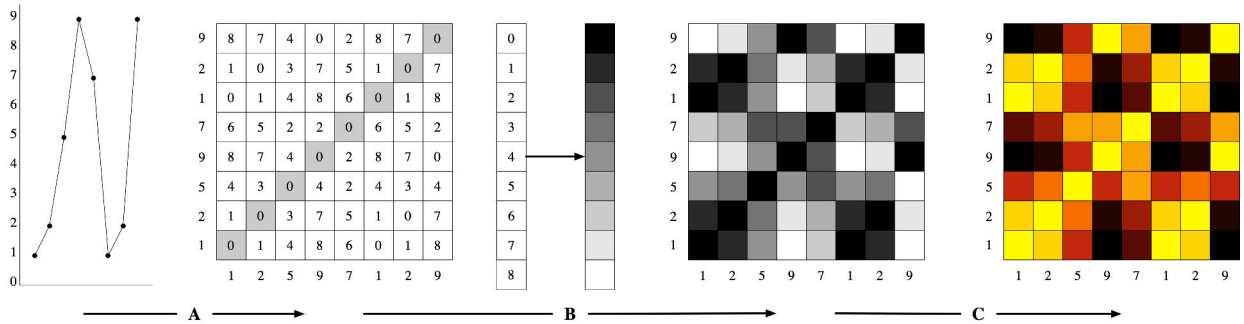


Figure 2. The steps involved in producing a self-similarity matrix.

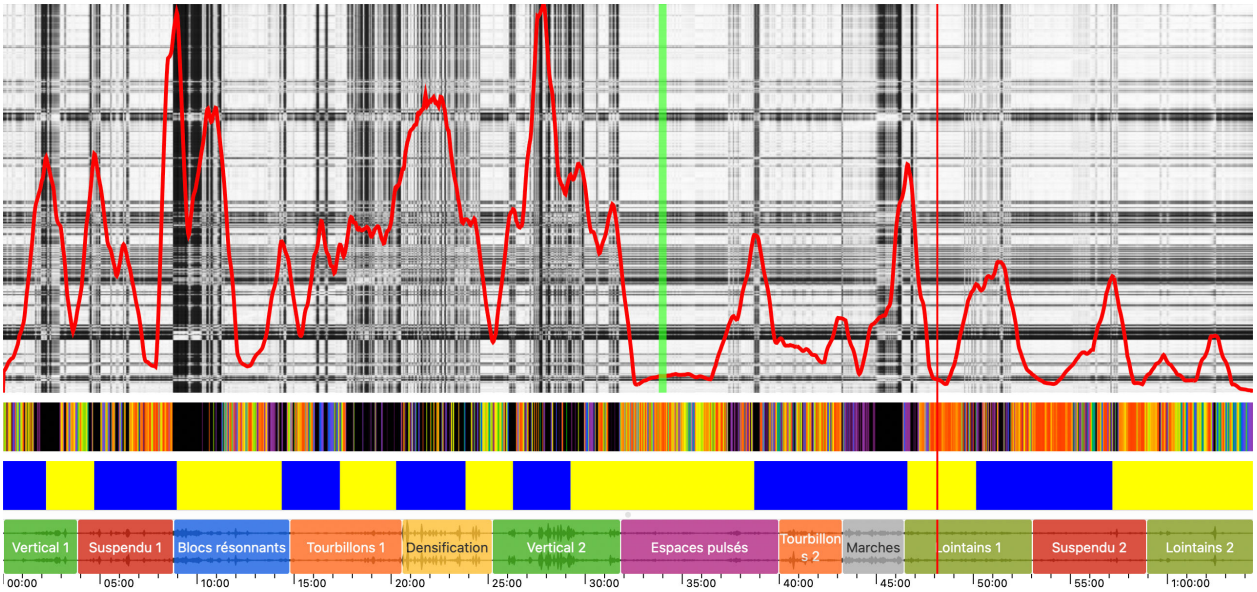


Figure 4. The interface for assisting the reading of a self-similarity matrix for the study of musical form in *Burning bright* by Hugues Dufourt.

Self-similarity matrices are particularly relevant in the analysis of musical form or microstructures[12] [23, 24, 25, 26, 27]. As true macroscopies[13], they also facilitate the exploration or navigation of long audio files. However, interpretation is difficult for the musicologist, which is why we have developed an interpretation-assistance system. This consists of 3 representations. The first one (top of figure 4) includes a self-similarity matrix computed from the FFT or from audio descriptors as well as a curve representing the novelty parameter. From this parameter, a segmentation is extracted (yellow and blue graph) using the technique of Foote and Cooper [30] based on peak recognition. A playback head (green) can be moved to display the result of a distance computation just below it (warm colors represent small distances). Finally, a main playback head (red) is used to navigate through the audio playback. The work represented in this figure is *Burning bright* by Hugues Dufourt. This is a 64-minute piece for percussion composed in one movement. Figure 4 also presents the segmentation (at the bottom) into tracks proposed in the recording. The aim is to facilitate the analysis of the musical form, which is difficult to grasp over such a long duration, but also to study concordances with the segmentation selected for the disc edition by the musicians and composer.

This interface is only in an experimental phase, but makes it possible to consider improvements for the analysis of long works or collections of works.

4. CONCLUSION

These three types of representation highlight what we underlined in the first part of the article. They are based on visualizations quite common in musical acoustics and their adaptation for the study of music makes them suitable tools for musicology. The development of these new methods is not based on a simple transfer but on the adoption of techniques specific to one scientific domain by another. It is in this sense that we have conceived our research into the study of 20th and 21st century music. This research obviously requires interdisciplinary skills, and these are indispensable to achieve as close a study as possible of a complex object such as music.

[12] The application of a similarity matrix for the comparison of two different versions of a work is described by Laura Zattra and Nicola Orio on Stria by John Chowning [26].

[13] The term macroscopy, in reference to the macroscope [27], is used here to designate representations offering a global view of the information extracted from the signal while keeping a trace of the details.

5. REFERENCES

[1] R. Bader, Springer Handbook of Systematic Musicology. Springer, 2018.

[2] N. Cook, E. Clarke, "Introduction : What is Empirical Musicology?," in Empirical musicology Aims, methods, prospects, Oxford University Press, 2004, pp. 3- 14.

[3] P. Couprie, "Methods and Tools for Transcribing Electroacoustic Music," in Proc. of the International Conference on Technologies for Music Notation and Representation, Montréal, 2018, http://tenor-conference.org/proceedings/2018/02_Couprie_tenor18.pdf.

[4] M. Castellengo, Ecoute musicale et acoustique. Eyrolles, 2015.

[5] P. Schaeffer, Traité des objets musicaux. Seuil, 1966.

[6] D. Besson, Les musicographies. INA-GRM/38es Rugissants/Mois du graphisme d'Échirolles, 1995.

[7] K. Stockhausen, No 3 Elektronische Studien Studie II. Universal Edition, 1997.

[8] A. Dall'Ara-Majek, "Symbolism in Acousmatic Music through an Analysis of La condition captive by Christine Groult," eOREMA 2, 2014, http://www.orema.dmu.ac.uk/?q=content/symbolism-acousmatic-music-through-analysis-la-condition-captive-christine-groult.

[9] E. Leipp, Acoustique et musique. Mines ParisTech, 2010[1971].

[10] R. Cogan, New Images of Musical Sound. Harvard University Press, 1984.

[11] M. Brech, Analyse elektroakustischer Musik mit Hilfe von Sonagrammen. Peter Lang, 1994.

[12] M. Helmuth, "Multidimensional Representation of Electroacoustic Music," Journal of New Music research 25(1), 1996, pp. 77-103.

[13] M. Clarke, "Wind Chimes: An Interactive Aural Analysis," in Portrait polychrome Denis Smalley, INA-GRM, 2010, pp. 35-57.

[14] S. Waters, T. Ungvary, "The Sonogram: A Tool for Visual Documentation of Musical Structure," in Proc. of the International Computer Music Conference, Glasgow, 1990, pp. 159-162.

[15] C. Roads, Musical Signal Processing. Routledge, 1997.

[16] X. Serra et al.. Roadmap for Music Information Research. 2013, http://hdl.handle.net/10230/21766.

[17] C. Seeger, "Toward a Universal Music Sound-Writing for Musicology," Journal of the International Folk Music Council 9, 1957, pp. 63-66.

[18] K. Neubarth, M. Bergeron, D. Conklin, "Associations Between Musicology and Music Information Retrieval," in Proc. of the International Society for Music Information Retrieval Conference, Miami, 2011, pp. 429–434.

[19] T. H. Park, Z. Li, W. Wu, "EASY does it: The Electro-Acoustic music analYsis toolbox," in Proc. International Society for Music Information Retrieval Conference, 2009, pp. 693-698.

[20] P. Couprie, "iAnalyse 5 : le développement d'un logiciel pour la musicologie numérique," in Journées d'informatique musicale, Bayonne 2019, https://jim2019.sciencesconf.org/resource/page/id/3.

[21] P. Couprie, "EAnalysis: Developing a Sound Based Music Analytical Tool," in Expanding the Horizon of Electroacoustic Music Analysis, Cambridge University Press, 2016, pp. 170-194.

[22] M. Malt, E. Jourdan, "Le BStD – une représentation graphique de la brillance et de l'écart type spectral, comme possible représentation de l'évolution du timbre sonore," in L'analyse musicale aujourd'hui, Delatour, 2015, pp. 111- 128.

[23] J. Foote, M. Cooper, "Visualizing Musical Structure and Rhythm via Self- Similarity," in Proc. of the International Computer Music Conference, Havana, 2001, pp. 419-422.

[24] J. Paulus, M. Müller, A. Klapuri, "Audio-Based Music Structure Analysis," in Proc. of the International Society for Music Information Retrieval Conference, Utrecht, 2010, pp. 625–636.

[25] G. Peeters, X. Rodet, "Signal-based Music Structure Discovery for Audio Summary Generation," in Proc. International Computer Music Conference, Singapore, 2003.

[26] M. Ramstrum, "Philippe Manoury's Opera K...," in Analytical Methods of Electroacoustic Music, Routledge, 2006, pp. 239-274.

[27] P. Couprie, "Analytical Approaches to Taurhiphanie and Voyage absolu des Unari vers Andromède by Iannis Xenakis," in From Xenakis's UPIC to Graphic Notation Today, Hatje Cantz Verlag/ZKM, 2020, pp. 434-456.

[28] L. Zattra, N. Orio, "ACAME – Analyse comparative automatique de la musique élec- troacoustique," Musimédiane 4, 2006, http://www.musimediane.com /numero4/LZattra/index.html.

[29] J. de Rosnay, The macroscope: A new world scientific system, Harper & Row, 1979.

[30] J. T. Foote M. L. Cooper, M. L., "Media segmentation using self-similarity decomposition," in Storage and Retrieval for Media Databases 5021, 2003, pp. 167-175.

ANNOTATING SYMBOLIC TEXTURE IN PIANO MUSIC: A FORMAL SYNTAX

Louis Couturier
MIS, Université de Picardie
Jules Verne, France
louis.couturier@u-picardie.fr

Louis Bigo
CRIStAL, UMR 9189 CNRS,
Université de Lille, France
louis.bigo@univ-lille.fr

Florence Levé
MIS, Université de Picardie
Jules Verne, France
CRIStAL, UMR 9189 CNRS,
Université de Lille, France
florence.leve@u-picardie.fr

ABSTRACT

Symbolic texture describes how sounding components are organized in a musical score. Along with other high-level musical components such as melody, harmony or rhythm, symbolic texture has a significant impact on the structure and the style of a musical piece. In this article, we present a syntax to describe compositional texture in the specific case of Western classical piano music. The syntax is expressive and flexible, unifying into a single text label information about density, diversity, musical function and note relationships in distinct textural units. The formal definition of the syntax enables its parsing and computational processing, opening promising perspectives in computer-aided music analysis and composition. We provide an implementation to parse and manipulate textural labels as well as a bestiary of annotated examples of textural configurations.

1. INTRODUCTION

Texture commonly refers to a high-level feature of music which strongly relates to musical style and form, and to the auditory perception of the musical flow. In the context of this work, as in symbolic music analysis in general, texture describes how sounding components (voices, chords, notes) are organized. *Variations* in classical music are examples of compositional practices in which a given melody and an underlying harmony are realized or arranged through different possible symbolic textures. Song covers in popular music are also intuitive examples in which an original texture is changed by another one while conserving the original chords and melody. Although essential in musical scores, symbolic texture appears to be challenging to formalize, probably due to the variety of musical concepts it gathers. This study aims at proposing such formalization through a well-defined syntax for the specific case of Western classical piano music, which opens perspectives for automated prediction of texture.

Copyright: © 2022 Louis Couturier et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Musicological studies of symbolic texture. In the musicological field, the description of the textural dimension for Western classical music is rather recent, compared for example to the study of harmony that has been developed from Rameau's treatise in 1722 [1]. The notion of texture was mentioned in the 1920s as an essential element to the study of form in non tonal compositions [2]. It has been developed in parallel to the analysis of contemporary music through the 20th century, with analogies with fine art paintings analysis (see [3] for usage of textural references by composers of late 19th and 20th). The term texture was only introduced in 1980 in the New Grove Dictionary of Music and Musicians, as *referring to the vertical aspect of musical structure [. . .] with regards to the way in which individual parts or voices are put together*. Berry [4] considers texture as one of the *structural function* in music, and several works have assessed its impact on the perception of structure [5, 6]. When referring to the overall organization of instrumental or vocal parts, some compositions can be categorized into common textural types like monophony, homophony or polyphony [5, 7, 8].

The principle of texture is closely related to timbre and register, and those dimensions particularly arise when considering orchestral texture [9–11]. In the case of piano music of the early 19th century, Hérold [12] dissociates the *textural factors of timbre*, as for example rhythmic or harmonic (horizontal or vertical) density, from its *pianistic factors*, which are more specific to the characteristics of the instrument.

Finally, a texture-based compositional approach can be found in the work of Moreira [13]. Textural configurations are thus described formally in the case of layered texture by stacking distinct textural parts. Different levels of abstraction and complexity are used, from ordered or unordered partitions of textural parts to more global categorizations in combinations of block and line elements.

MIR studies of symbolic texture. In the Music Information Retrieval field (MIR), the notion of texture is mostly studied in the audio domain where it is commonly associated with timbral properties [14] for tasks such as genre classification [15, 16].

In the symbolic domain, works on texture identification are still sparse, probably due to the difficulty to formalize this notion. However, a few attempts have been made,

restricting the study to a specific repertoire. A machine learning approach has been used for the classification of texture in Italian madrigals of the 16th century [17]. On classical string quartets, a syntax has been proposed, highlighting for each instrument its role (melody or accompaniment) and its relations to other voices [18, 19]. Identifying musical roles through texture analysis has also been done for the specific case of guitar popular music, using audio [20] and symbolic data [21]. Detecting textural changes was also shown to be a promising approach to delimit successive sections of a piece in music structure retrieval [22]. Alternatively, the formalization of some selected textural figures has been used to synthetically increase the training set of a machine learning model dedicated to roman numeral analysis [23].

Beyond musical analysis, the recent growth of the music generation field has also contributed to bring new challenges in symbolic texture modeling. This is the case for music style transfer where the texture of a first musical piece, commonly assimilated to its *style*, is mapped on the harmony and melody of a second one [24]. The separation of harmony and texture is also found in [25], where these two dimensions are disentangled in a latent space for more controllable music generation.

Motivation and outline. We believe that texture analysis can improve our understanding of compositional practice by helping the study of both musical style and musical structure. Such improvements could benefit to the field of music generation where these high level musical layers are known to be difficult to control.

This work introduces a syntax to annotate texture for symbolic piano music. In comparison to texture analysis in other repertoires, such as classical string quartets [18] in which each instrument mostly plays one voice at a time, a major challenge of piano music is the complex organization and high variability of voices and streams during the piece. Here, we focus on the specificities of piano music of Viennese classicism, whose major composers were Haydn, Mozart and Beethoven. By doing so, we ensure consistency and possibility of comparison with previous work on string quartets.

The contributions of this work are as follows. We identify the major characteristics of symbolic texture that are relevant for classical piano music based on existing studies (Section 2). In Section 3, we define a formal syntax to facilitate the computational manipulation of these characteristics. Finally, Section 4 presents a collection of annotated examples.

2. TEXTURAL CHARACTERISTICS

Based on musical and musicological resources, complementary aspects of texture have been taken into account in our study to describe the texture of a score region. Selected characteristics are detailed in this section and illustrated in Figure 1.

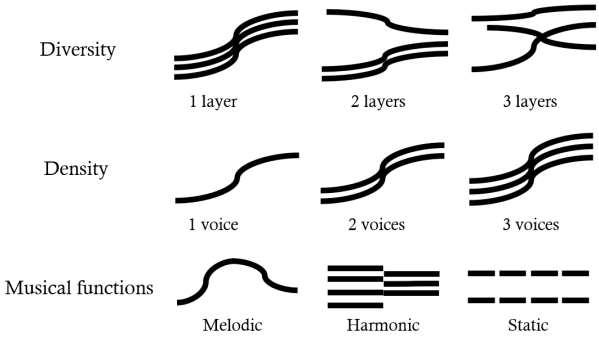


Figure 1. Schematic representation of main textural characteristics. Diversity: three voices to form respectively one, two or three layers. Density: a unique layer of density 1, 2 or 3. Musical functions: melodic, harmonic, static. Additional elements are used to qualify the internal organization of each layer, to indicate characteristic musical figures or specific relationships between their voices.

2.1 Diversity and Density

The analysis of texture is related to the organization of atomic musical components – i.e. notes – into perceptually coherent textural units. According to Huron [7], *diversity* and *density* respectively qualify the quantity and the (in)homogeneity of sounding components in an auditory scene. Similarly, Moreira [13] defines a textural configuration by *the organization of n threads (quantitative aspect) into m textural parts (qualitative aspect) in a given passage of music*. In our work, these textural parts are referred to as *layers*. One layer can be made of either one unique voice, as for example in the case of a single melodic line, or several voices gathered by some kind of similarity, as for example voices in homorhythmy (sharing the same rhythm). Three-voice inventions by Bach are typically composed of 3 single-voice layers (similarly to the top right representation in Figure 1), whereas a chorale is more likely to be made of one unique 4-voice layer, due in this case to the homorhythmy of the voices. Note that homorhythmic voices are not necessarily perceived as a single layer for example if their respective register, motion or articulation make them differ enough. Algorithms to group notes in voices or coherent groups (streams) have been proposed in [26–28] but without specifically qualifying the texture of those groups.

2.1.1 Distinct Layers (Diversity)

The perception of distinct textural layers can emerge from several grouping mechanisms described in the works of McAdams and Bregman [29, 30], including both vertical and horizontal similarities (simultaneity and sequentiality). Onset-synchrony and semblant motions are core criteria for the fusion of notes or voices into layers, as highlighted in [7]. The number of distinct layers (named cardinality in [13]) corresponds to the *diversity* of the texture. In our work, based on these considerations, a primary step is to identify the distinct layers that would be most plausibly perceived as independent by a human listener. Note that

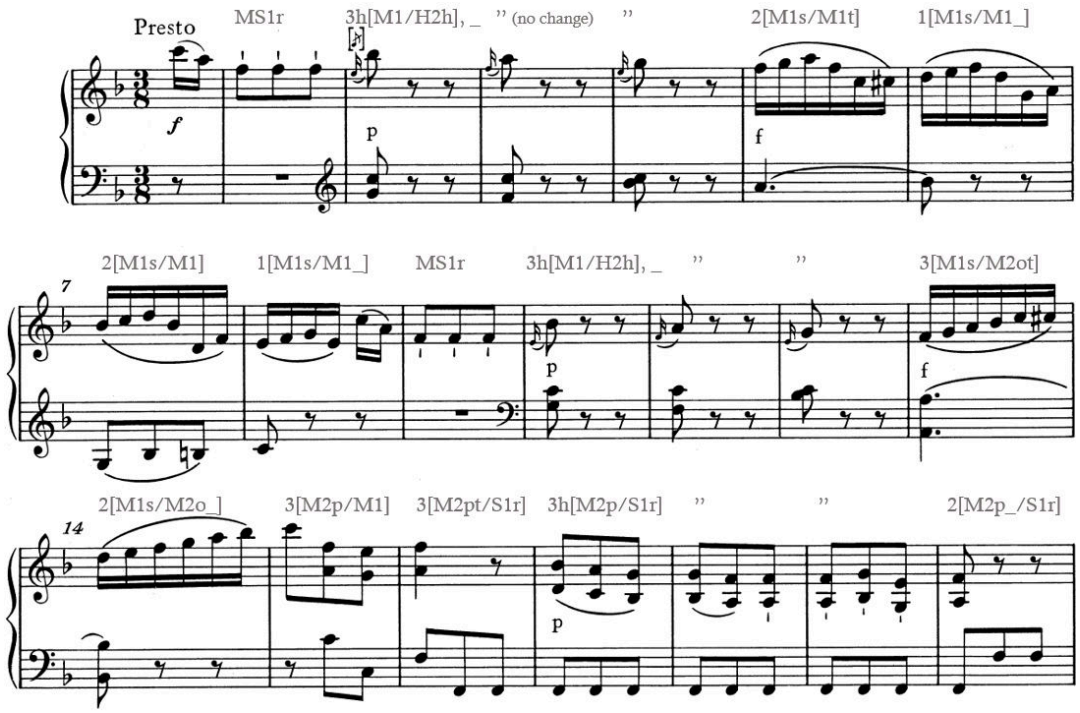



Figure 2. First bars of the *Presto* from Piano Sonata n°2 by W. A. Mozart (K. 280/189^e), annotated with texture labels using proposed syntax. Textural labels are given in their detailed expression which begins with their global density value. However, this information can be omitted when the global density is equal to the sum of the density of each layer. The label of bar 5 can be shortened this way: `M1s/M1t` instead of `2[M1s/M1t]`. On the contrary, `1[M1s/M1_]` in bar 6 cannot.

we will not directly refer to the number of distinct layers, we will simply describe them one by one.

2.1.2 Number of Voices (Density)

The (vertical) *density* of a layer (named thickness in [13]) corresponds to the number of simultaneous notes heard at the same time in this layer. When considering a musical fragment with several concurrent layers, the *global density* of this fragment can be considered in most cases as equal to the sum of the densities of the constituting layers, as it is done in Moreira's textural spaces [13] or Gentil-Nunes Partitional Analysis [31]. But in some more complex textural configurations, we allow for the global density to take a value different from the sum of the densities of its constituting layers. For example, the alternation of two single-voice layers in call and response would lead to a global density value equal to one. In Figure 2, bar 1 has a global density of 1 (1 layer of density 1), bar 5 has a global density of 2 (2 layers of density 1), whereas bar 6 has a global density of 1 (since the bass layer is very short compared to the upper layer).

2.2 Melodic/Harmonic/Static Functions

In [18], the distinction is made between melodies and accompaniment parts. Instead of characterizing such roles in the overall textural configuration, we propose to focus on the functional aspect of each layer taken individually, expressed by a combination of three labels: *melodic*, *harmonic* and *static*. These terms are named based on Benward and Saker's primary elements of musical texture [8]. These three complementary dimensions respectively convey ideas of horizontality, verticality and stability.

Melodic A layer labeled as *melodic* (`M`) is shaped as a sequence of notes that expresses a strong sense of temporal continuity. It is not limited to dominant phrases. The scope of melodic layers includes primary, secondary or supporting melodies in Benward and Saker's primary elements [8]. Counterpoint only has `M`-function layers, each voice existing independently. In most cases, a bass line could also be considered as an `M` layer. Furthermore, a melodic-function layer is not necessarily monodic – that is to say, with a density of one. It can be a sequence of harmonic thirds, sixths or even denser chords that form a melodic continuity. In this last case, it would additionally be labeled as *harmonic*. In the extract of Figure 2, the upper layer has systematically a melodic function and the left hand sometimes also has a melodic function, as from bar 5 to bar 8.

Harmonic *Harmonic* layers (`H`) qualify harmony-related layouts, such as chords or broken chords, which raise generally consonant combination of notes. For example, in bars 2 to 4 in Figure 2, the left hand plays a two-voice harmonic layer. Note that the density of a harmonic layer can be equal to one, as in the case of arpeggios.

Static Derived from Benward and Saker's *static supporting parts* [8], our *static* label (`S`) encompasses layers struc-

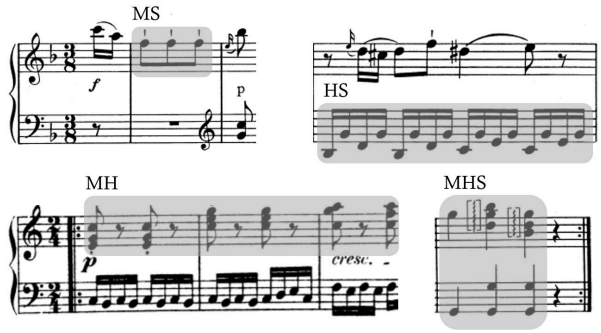


Figure 3. Example of textural layers with combined melodic M, harmonic H and static S musical functions. From Mozart K.280/189[e] mm. 1-2; K.279/189[d] mm. 6, K.265/300[e] variation 6 mm. 1-3 and K.309/284[b], mm. 58.

tured with short-term repetition or sustained notes, like pedal notes or drones. Repeated notes might not necessarily be consecutive: ostinati are typically considered as static layers for they repeat short cells and are characterized by regularity or persistancy. In Figure 2, the bass layer in bars 16 to 20 is static, since it is constituted of a single repeated note. Although this concept is rarely mentioned in the literature, it has shown to be helpful in our preliminary annotation experiments.

Mixture of functions Mutiple functions can apply to a single layer. Common examples of each combination are given in Figure 3.

- (MS) A melody with short term repetition, expressing both continuity in the voicing and insistance on a given pitch, could have both melodic and static function;
- (MH) A chordal melody would have both melodic and harmonic function;
- (HS) A repeated Alberti bass, emphasizing harmony with a repetitive figure, would have both harmonic and static function, as would have a triple hammer blow [32] on a given chord;
- (MHS) In rare cases, a layer can involve at the same time the three dimensions M, H, and S. For example, a hammer blow which does not repeat exactly the same chord, but has a changing upper note, involves some melodic movement.

2.3 Internal Organization of a Layer

2.3.1 Voice Relationships within a Layer

In the case where several voices are gathered in the same layer because of similar rhythm or semblant motion, it appears important to specify which degree of similarity they share. Inspired by terminology and description of [18], we use three levels of description between voices of a single layer.

- Homorhythmy (h): all the voices or notes in the layer share a strongly similar rhythm. In the bars 2 to 4 in Figure 2, the left-hand harmonic part has its two voices played simultaneously, in homorhythmy.
- Parallel motions (p): all the voices in the layer share (almost) the same rhythm and the same intervallic consecutive motions. With third or sixth doubling, for example, major or minor intervals are tolerated. See the melodic layer in measures 17, 18 and 19 in Figure 2.
- Octave motions (o): all the voices in the layer have parallel motions plus the same pitch classes. In Figure 2, bars 13-14, the left-hand echoes the motion in bars 5-6, this time with octave doubling.

Note that octave motions are a specific case of parallel motions, which are also a subcase of homorhythmy. Because of these inclusive relations, we consider the most restrictive attribute to systematically induce the included ones. Single voice layers are naturally not concerned by these relations.

2.3.2 Characteristic Musical Figures

Each layer can be associated with specific attributes to precise how it is written. We defined some characteristic figures that seem relevant to the studied repertoire:

- sustained notes (t): the notes in the layer are sustained during most of a musical segment (see the left hand part in bar 5 and 13 in Figure 2).
- repeated notes (r): short-term repetition of the exact same pitches one after another. Each new pitch in the layer is repeated at least once (Figure 2, bars 16 to 20).
- sparse (_): layers with low horizontal density, locally. On an indicative basis, less than half time of studied section is filled with the notes of the layer (Figure 2, bars 2 to 4).
- scale (s): ascending or descending sequence of short duration notes (Figure 2, bars 13 and 14). Repeated ascending or descending short motives can also be considered (Figure 2, bars 5 to 7).
- oscillation (b): fast alternation between two notes. An example can be found in the bestiary that is provided at the end of the paper, bars 38 and 41 to 43.

These last attributes tend to be more style-specific and are intentionally biased towards Western classical style, here. Room is left for future extensions.

2.4 Sublayer Decomposition

In addition to the decomposition of a texture into simultaneous layers, a single layer can be decomposed itself into several *sublayers*. A sublayer decomposition can be useful to indicate relationships between a subset of voices included in the same layer. In a homorhythmic chorale for

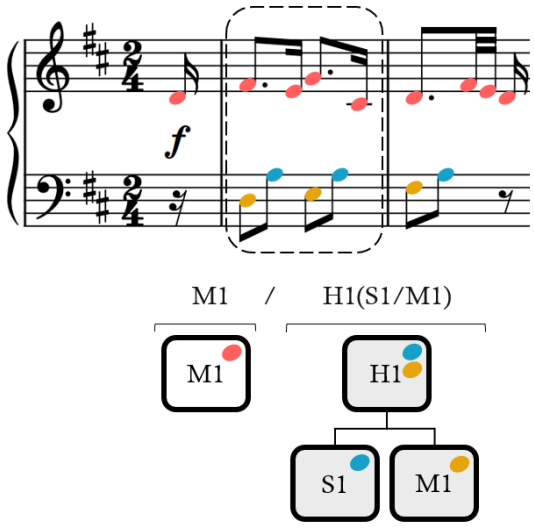


Figure 4. Textural annotation and tree structure of the first measure of the *Allegro moderato* from *Partita in D major for piano*, attributed to J. Haydn (Hob.XVI.14, 1767). The bar is labeled `2[M1/H1(S1/M1)]`. Two main layers with density of one are separated: a melodic (`M1`) and a harmonic (`H1`) part. Furthermore, the lower part can be decomposed into two sublayers: a melodic stream can be distinctly heard (`M1`: D, E, F sharp), in alternation with the repeated note A with static function (`S1`).

example, this would be done by splitting the top-level `MH` layer into multiple `M` sublayers. Sublayer decomposition can also be useful in case of compound melodies [33] or two-part accompaniments such as stride piano. In the latter case, it enables to describe bass notes and chord notes in two distinct sublayers.

Top-level textural layers are described rather independently, whereas sublayers are interconnected and help describing the content of a given single layer. The subdivision of layers into sublayers occurs recursively and handles vertical separation at different levels of precision. An example of sublayer decomposition is shown Figure 4.

3. A SYNTAX FOR TEXTURE ANNOTATION

In this section, we develop a general syntax for notating symbolic piano texture in flat text that is both musically intuitive and flexible but at the same time rigidly structured. The syntax enables to depict into a single text label the five characteristics of texture described in Section 2 and illustrated Figure 1.

3.1 Formal Structure

In this work, and similarly to [18], it is proposed to annotate texture at the bar level. We estimate that a description of texture with shorter time span would indeed limit the representation of high level textural concepts previously described. On the other hand, notating the texture of several consecutive bars by a single annotation might yield to labels that would be either overloaded in text or too approximative regarding the musical surface. It is not uncommon to see a drastic change of texture in the middle of a bar, for example at the transition between two different structural sections. To overcome such cases, the syntax enables to indicate two consecutive textures separated with a comma.

Topologically, we represent a textural configuration by a data structure of forest, i.e. a set of trees, in which each tree corresponds to a top-level layer. It transcribes the sublayer description in section 2.4 by splitting a layer description (a node) into the individual descriptions of their sublayers (children nodes). The depth in the tree represents a level of precision: roots of a textural tree portrays surface description of the top-level layers whereas leaves can detail a finer voice-specific description. An example is given in Figure 4.

3.2 Flat Text Syntax

The syntaxic formalism we propose is described in Backus-Naur form in Figure 5. A texture is either composed of a unique textural layer or a set of layers. In the latter case, the layers are ordered from high to low register and separated by the character `/`. Each layer is expressed individually by the combination of its musical functions (`M`, `H`, `S`, `MH`, `MS`, `HS`, `MHS` or `N` for None) and its density (as a positive integer value), followed by all the relevant attributes describing the internal organization of the layer: `M1`, `MH4h` or `S2ot` for example.

In case a sublayer decomposition is needed, it has to be inserted after each corresponding layer by defining the ordered set of sublayers in parenthesis: `A/B(`B_a`/`B_b`)`. Finally, an integer indicating the global density is added by encompassing the whole label with brackets `[]`. This notation is only necessary when the global density does not correspond to the sum of all layer densities: `M1/M1/S2` is a shortcut notation of `4[M1/M1/S2]`, which is perfectly equivalent. At this global level, additional tags may also be added, like in the second measure of Figure 2, `3h[M1/H2h]`, which links two separated layers with homorhythmy.

A *Python* implementation of this structure can be found at `http://algomus.fr/code`. It allows the parsing of the syntax by converting flat text texture labels into structured objects.

3.3 Influence of the Context

Even though texture is annotated at the granularity of the musical bar, it is worth mentioning that the context (i.e. neighbor bars) plays an important role in the identification of the local texture. We illustrate how it impacts the annotation by using three examples from Figure 2.

- *Pattern repetition*: the inclusion of the scale attribute (`s`) in bar 5 to 7 takes into account the repetition of the melodic cell on the three bars. The motive is consecutively transposed in a descending manner. Bar 5 alone is not sufficient to detect this pattern.
- *Layer continuity*: the parallel attribute (`p`) in the upper part of bar 16 results from a continuity of the

```
<bar-texture>      ::=  <texture> [',' <texture>]            # Label for a single bar
<texture>          ::=  <description> '[' <layer-list> ']'     # Detailed
                      | <layer-list>                           # Shortcut
                      | _                                      # Silence
<layer-list>       ::=  <layer> ['/' <layer-list>]
<layer>            ::=  <layer-function><description> ['(' <layer-list> ')']
<description>      ::=  <density> [<relation>][<attribute-set>]
<attribute-set>    ::=  <attribute> [<attribute-set>]
<layer-function>   ::=  M | H | S | MH | MS | HS | MHS | N
<relation>         ::=  h | p | o          # homorhythmy, parallel motions, octave motions
<attribute>        ::=  s | t | b | r | _   # scale, sustained, oscillation, repetition, sparse
<density>          ::=  .. integer in ℕ* ..
```

Figure 5. Syntax of Texture Notation in Backus Naur Form. As an example, the `2[M1/H1(S1/M1)]` label from Figure 4 contains a <layer-list> of two elements. The first <layer> (`M1`) has a melodic <layer-function> `M` and a <density> of one. The second <layer> has a nested <layer-list> `S1/M1`, to describe its sublayers. The value of 2 corresponds to the global <density> in the detailed <description>.

layer `M2p` (two-voice parallel melodic layer) started in bar 15.

- *Layer separation*: bars 2, 3 and 4 could have been annotated `MH3h` in another context due to the onset synchrony, which tends to merge notes into the same stream [7]. However, the choice has been made to annotate it with two distinct top-level layers, to preserve the continuity of the melodic part that has been heard just before. It assumes that the perception of streams is in this case more influenced by the sequential grouping of notes rather than vertical grouping. The homorhythmy is then indicated as a global attribute, resulting in the label `3h[M1/H2h]`.

4. ANNOTATED EXAMPLES

To accompany the presentation of our syntax, we provide a *Bestiary of musical textures* (see Figure 6), which aims at illustrating common examples of textural configurations.

The bestiary does not intend to provide an exhaustive description of all possible combinations of layers that can be found in classical piano music. Instead, it focuses on the illustration of typical layers. It constitutes not only a useful tool and reference to understand textural annotation, but also a catalog of textural ideas created to be extended in the future.

In addition to the synthetic fragments gathered in the bestiary, the syntax was experimented on musical extracts from the classical piano repertoire. The first measures of the third movement of Mozart Piano Sonata n°2 (K. 280/189e) are shown in Figure 2. The annotation of the full piece is available in text format at `http://algomus.fr/data`.

5. CONCLUSIONS

In this article, a new syntax has been defined and illustrated for the annotation of symbolic texture in piano music of the Western classical-era. Based on musicological reflexions, the textural elements we presented and their organization provide a practical framework for texture analysis. By integrating them in a unified syntax, we facilitate their computational manipulation for analysis and generation tasks.

Future works include the elaboration of a distance function that could systematically be applied to evaluate the similarity between two textural labels. The similarity between two labels should reflect the similarity between the corresponding symbolic textures. Besides texture analyses and comparison between composers or eras, this project also aims at studying the evolution of texture within a single piece of music. Strong links are to be found between texture and the structure of the piece, for instance in the case of sonata form. Future work will also include the creation of a consistent dataset, along with guidelines to annotate texture in practice. Resulting data would be usable for the training of supervised models for automatic texture estimation in classical piano scores. Additionally, our texture model is a step towards the interpretation of arbitrary texture categories that would be discovered using unsupervized methods. It would finally contribute to further improve the understanding and the control of this musical dimension in generative tasks.

Acknowledgments

This work has been partly funded by Région Hauts-de-France and CPER MAuVE.

6. REFERENCES

[1] J.-P. Rameau, *Traité de l'Harmonie réduite à ses principes naturels*, 1722.

[2] C. P. d. Silva, "Por uma definição unificada de textura musical," Master's thesis, Escola de Música, Universidade Federal da Bahia, 2018.

[3] S. Perraudeau, "La texture en musique: sa contribution pour la composition, l'apprentissage de la musique et ses effets sur la perception musicale et la cognition des enfants sourds implantés," Ph.D. dissertation, Université Bourgogne Franche-Comté, 2019.

[4] W. Berry, *Structural functions in music: A Probing Exploration of Conceptual Bases and Techniques for the Analysis of Tonality, Texture, and Rhythm.* Prentice-Hall, 1976, reprint New York, Dover, 1987.

[5] J. M. Levy, "Texture as a sign in classic and early romantic music," *Journal of the American Musicological Society*, vol. 35, no. 3, pp. 482–531, 1982.

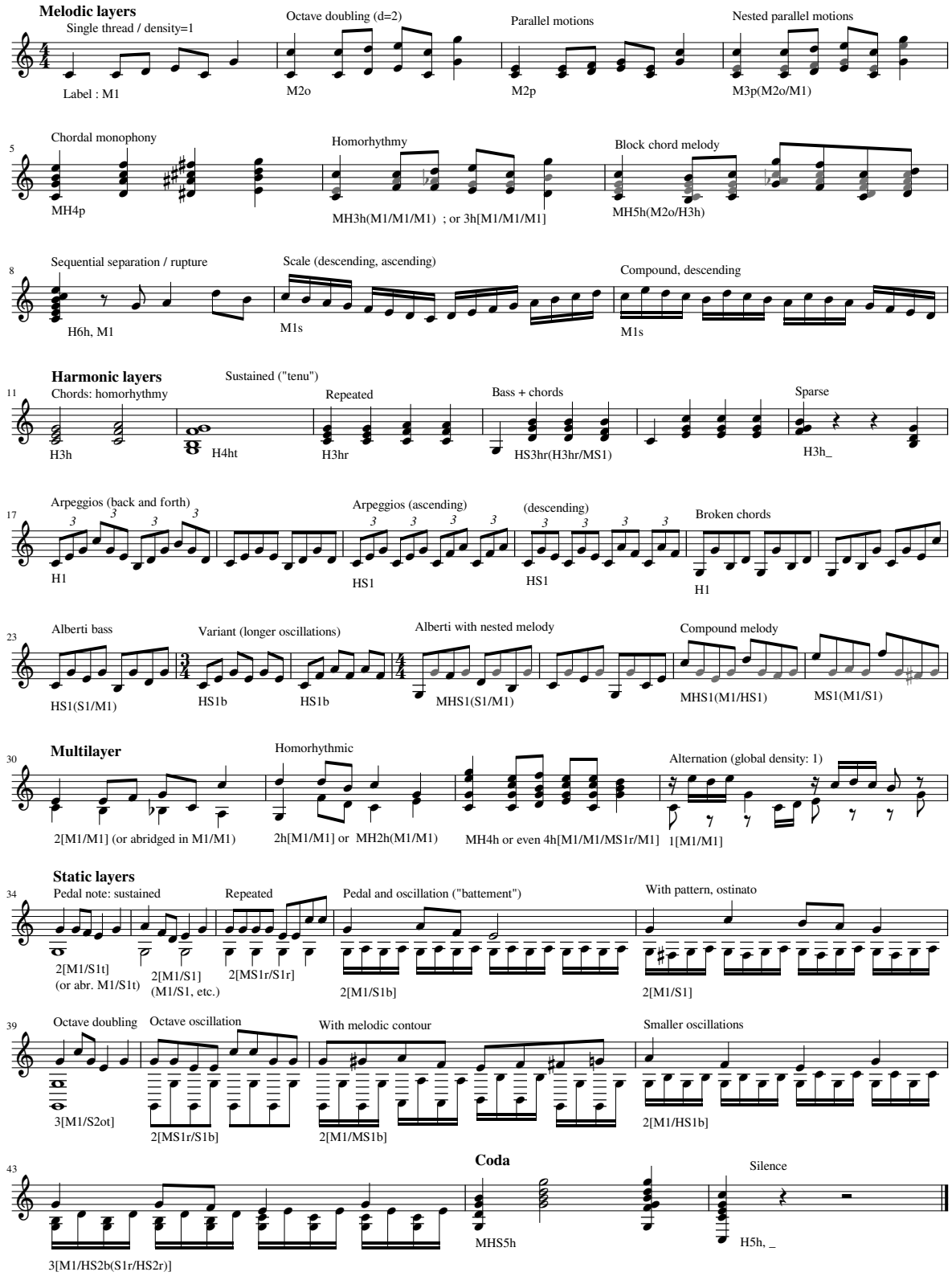



Figure 6. *Bestiary of musical textures*. The texture of each bar is annotated with the syntax defined in Figure 5, illustrating various textural layers and combination of layers that could be found in piano music. The file is available at `http://algomus.fr/data`.

[6] B. Duane, "Texture in eighteenth- and early nineteenth-century string-quartet expositions," Ph.D. dissertation, Northwestern University, 2012.

[7] D. Huron, "Characterizing musical textures," in *International Computer Music Conference (ICMC)*, 1989, pp. 131–134.

[8] Bruce Benward, Marilyn Saker, *Music: In Theory and Practice, Vol. I. Eigth Edition.* McGraw-Hill, 2008, ch. 7, p. 145–162.

[9] W. Piston, *Orchestration*. WW Norton, 1955.

[10] Q. R. Nordgren, "A measure of textural patterns and strengths," *Journal of Music Theory*, vol. 4, no. 1, pp. 19–31, 1960.

[11] D. Guigue and C. de Paiva Santana, "The structural function of musical texture: Towards a computer-assisted analysis of orchestration," in *Journées d'Informatique Musicale (JIM)*, 2018.

[12] N. Hérold, "Timbre et forme : La dimension timbrique de la forme dans la musique pour piano de la première moitié du dix-neuvième siècle," Ph.D. dissertation, 2011.

[13] D. Moreira, "Textural design: a compositional theory for the organization of musical texture," Ph.D. dissertation, Centro de Letras e Artes, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro), 2019.

[14] G. Strobl, G. Eckel, and D. Rocchesso, "Sound texture modeling: A survey," in *Sound Music Computing*, 2006.

[15] L. Nanni, Y. Costa, and S. Brahnam, "Set of texture descriptors for music genre classification," in *22nd International Conference in Central Europeon Computer Graphics, Visualization and Computer Vision (WSCG)*. Václav Skala-UNION Agency, 2014.

[16] J. H. Foleis and T. F. Tavares, "Texture selection for automatic music genre classification," *Applied Soft Computing*, vol. 89, pp. 106–127, 2020.

[17] E. Parada-Cabaleiro, M. Schmitt, A. Batliner, B. Schuller, and M. Schedl, "Automatic recognition of texture in renaissance music," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[18] M. Giraud, F. Levé, F. Mercier, M. Rigaudière, and D. Thorez, "Towards modeling texture in symbolic data," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 59–64.

[19] L. Soum-Fontez, M. Giraud, N. Guiomard-Kagan, and F. Levé, "Symbolic textural features and melody/accompaniment detection in string quartets," in *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2021.

[20] R. Foulon, P. Roy, and F. Pachet, "Automatic classification of guitar playing modes," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2013, pp. 58–71.

[21] D. Régnier, N. Martin, and L. Bigo, "Identification of rhythm guitar sections in symbolic tablatures," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[22] A. Tenkanen and F. Gualda, "Detecting changes in musical texture," in *International Workshop on Machine Learning and Music*, 2008.

[23] N. N. López, M. Gotham, and I. Fujinaga, "Augmentednet: A roman numeral analysis network with synthetic training examples and additional tonal tasks," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[24] O. Cífka, U. Şimşekli, and G. Richard, "Groove2Groove: one-shot music style transfer with supervision from synthetic data," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2638–2650, 2020.

[25] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[26] E. Cambouropoulos, "Voice and stream: Perceptual and computational modeling of voice separation," *Music Perception*, vol. 26, no. 1, pp. 75–94, 09 2008.

[27] D. Makris, I. Karydis, and E. Cambouropoulos, "Visa3: Refining the voice integration/segregation algorithm," in *Proceedings of the Sound and Music Computing Conference (SMC)*, 2016.

[28] E. Chew and X. Wu, "Separating voices in polyphonic music: A contig mapping approach," in *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2005, pp. 1–20.

[29] S. McAdams and A. Bregman, "Hearing musical streams," *Computer Music Journal*, vol. 3, no. 4, pp. 26–60, 1979.

[30] A. S. Bregman, *Auditory scene analysis: The perceptual organization of sound*. MIT Press, 1994.

[31] P. Gentil-Nunes, "Análise particional: uma mediação entre composição musical e a teoria das partições," Ph.D. dissertation, Centro de Letras e Artes, Universidade Federal do Estado do Rio de Janeiro, 2009.

[32] J. Hepokoski and W. Darcy, "The medial caesura and its role in the eighteenth-century sonata exposition," *Music Theory Spectrum*, vol. 19, no. 2, pp. 115–154, 1997.

[33] J. Lester, *Compositional theory in the eighteenth century*. Harvard University Press, 1992.

Exploring the Possibilities of Music Therapy in Virtual Reality: Social Skills Training for Adolescents with Autism Spectrum Disorder

Linnea Bjerregaard Pedersen
Aalborg University, Copenhagen
lbpe17@student.aau.dk

Ali Adjorlu
Aalborg University, Copenhagen
adj@create.aau.dk

Valentin Bauer
Paris-Saclay University
valentin.bauer@limsi.fr

ABSTRACT

Autism spectrum disorder (ASD) is a neurodevelopmental condition characterised by profound communication and social interaction deficits. It can lead to social anxiety and depressive thought patterns, leaving the individual unable to be a productive member of society. Music therapy is a method that applies musical experiences and the relationships that develop through the sessions to train communication skills, thus addressing one of the core difficulties of ASD. However, music therapy requires musical instruments that are not always easily accessible and can be an anxiety-inducing experience. This paper presents a multiplayer virtual reality application designed to allow a music therapist to help a child develop social skills in a virtual environment through musical activities. The application is designed in collaboration with two music therapists and two psychologists and includes a qualitative evaluation of the application.

1. INTRODUCTION

Autism Spectrum Disorder (ASD) is a neurodevelopmental condition that is mainly characterised by social communication and interaction difficulties [1]. These deficits can lead to social anxiety [2], isolation [3], and depression [4]. Music therapy (MT) has shown potential in increasing social communications skills in individuals on the autism spectrum [1]. More specifically, MT successfully trains verbal and non-verbal communication, especially if the intervention starts during the childhood of autistic individuals [7].

Compared to verbal stimuli, music-based interventions can increase the child's engagement, attention, as well as their self-esteem [8–10]. However, MT can be a socially anxiety-inducing experience for autistic children [2], as well as time-consuming for the parents, thus risks discouraging families to continue the MT sessions due to not seeing immediate progress [7].

By replacing the users' physical sensory information with virtually created ones, Virtual Reality (VR) can be used to place the users inside a relevant, interactive virtual environment (VE) within which they can receive appropriate interventions [11]. This enables the opportunity to perform MT from the social safety of the user's home and without the need for resources to access physical MT sessions and instruments. So far, most research on VR interventions for autism has focused on training target behaviours through simulated real-life situations [12, 13], including costly or potentially threatening scenarios [14].

However, the potential of VR to extend MT sessions for autistic children by alleviating social interaction difficulties [15], remains largely unexplored. To the authors' knowledge only few VR MT studies have been performed so far [16–18]. Shahab et al. [16] tested a VR environment consisting of a robot avatar and a virtual xylophone with five children, suggesting promising outcomes regarding musical ability and cognitive skills. Bryce et al. [17] exposed four children to a 360° video of a children's choir, but findings were inconclusive. Brungardt et al. [18] conducted a pilot study with 17 patients in palliative care. The intervention received positive feedback and was accepted and found usable by the patients.

When using VR applications for individuals with ASD, approaches have to be individualised according to the specific needs, to increase the overall acceptability [17, 19]. To that respect, involving stakeholders in the design process while offering collaborative designs is advised [19].

This project follows the philosophy of user-centered design [20], meaning that therapists working with autistic individuals on a daily basis are included early in the design process. In this paper, we present the design process and validation of a collaborative MT multiplayer platform that aims at allowing music therapists to perform MT sessions with an autistic child in VR. The overall design process is divided into four segments, which are successively described in the following sections. First, to create the initial design of the VE, an informal preliminary interview was conducted with two music therapists. (section 2). Second, one of the music therapists who participated in the preliminary interview took part in the evaluation of this first design version (section 3). Third, frequent meetings were organised with two psychologists over four consecutive months, to further refine the design and validate the possibility to conduct future testing with them (section 4). All interviews were conducted remotely with respect to the COVID-19 regulations. At last, a pre-test was orchestrated with 25 non-autistic children to validate the possibility to use the developed VR platform in the two real-world clini-

[1] 'Autistic person' and 'person on the autism spectrum' are endorsed terms by autistic adults and stakeholders, i.e., parents [5, 6].

Copyright: © 2022 Linnea Bjerregaard Pedersen et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

cal settings evoked above with autistic children, i.e., music therapists' and psychologists' sessions (section 5).

2. INITIAL DESIGN PROCESS WITH TWO MUSIC THERAPISTS

2.1 Method for the First Design Iteration

To acquire fundamental knowledge about designing for MT and children on the autism spectrum, initial design guidelines were derived from the literature. In particular, Baltaxe-Admony et al. [21] investigated the reasons behind the deficiency of technology in MT by interviewing six music therapists from around the world. When inquiring about what the therapists were missing from their previous experiences with technological solutions, five design considerations emerged: 1) versatility, 2) ease to travel with, 3) easy to set up, 4) a stand-alone technology, and 5) possibility for data collection. Points 2 to 4 led to choose the Head-Mounted Display (HMD) Oculus Quest 2 for developing the application, as it is wireless, portable, and has relative ease of setting up. Point 1, versatility, was included as a discussion topic for planned interviews. Point 5, data collection, was omitted due to being irrelevant for the time being.

Building on these design guidelines, an interview with two music therapists was conducted to understand their specific needs when working with autistic children. One had a 6-year education in the MT field and 13 years of professional experience, including a diverse clientele. Her typical sessions last around 45 minutes for children and 2 hours for adults. They usually take place at the client's home, and can be individual or group sessions. The second music therapist is a professor with vast experience in researching how MT can be used to help autistic children [22, 23]. The interview lasted about an hour. The researchers asked questions about the design of the VE, the interaction with the instruments, and the possibilities for training social and communication skills.

2.2 Findings from the Preliminary Interview

2.2.1 Designing the Music Activity

The music therapists first suggested making the VE familiar and simple. Thus a classroom was designed, as it is a common location for MT sessions. To make it resemble a typical classroom, items such as a bookcase, a blackboard, school lockers, and a row of school desk chairs were added to the VE.

The instruments should also be familiar while also taking advantage of VR and the ability to make elements that are otherwise not physically possible. Thus, two VR Musical Instruments (VRMIs) were created: a virtual xylophone as the familiar instrument, and *The Looper* as the specific VRMI (Figure 1). Their designs draw upon guidelines outlined by Serafin et al. [24]. In particular, both natural and 'magical' interaction possibilities were added, where the xylophone was promoting natural interaction and the Looper the 'magical'. Other guidelines were considered,

Figure 1. *A view of the classroom. In the front, the play area with the two instruments, the Looper (left) and the xylophone (right).*

such as using appropriate feedback and mapping, implementing for minimal latency, and preventing cybersickness [24]. Last but not least, the experience had to be social, which is also often highlighted in technology-based autism research [19]. To do so, the Looper was inspired by the social interaction occurring during board games. Hence, the therapist and child would both be interacting with the Looper across the table, thus prompting joint attention. At last, the therapist could construct rules such as turn-taking or other rules that can be customised to the child's needs.

The Looper consisted of an oval track with an RGB wave effect to make the instrument salient and attract the child's attention. A small black ball was looping the track, the speed of which could be controlled by adjusting a handle on the side of the Looper. When passing by one of the white dots on the track, a bass drum sound played, functioning as a backbeat. Then, the user could add cubes on the track with different sound modes. When the ball passed through the cubes, they would play a sound and provide visual feedback, consisting of a brief change in colour (white to yellow) and a simple particle system with pink sprinkles. The different sound modes were a snare drum, kick drum, crash cymbal, and a 'wobbling' sound. To support predictability, they are all indicated by fitting icons on the cubes.

To avoid cybersickness, it is advised to minimise the amount of acceleration while being mindful of the locomotion techniques available [24]; continuous locomotion techniques were implemented for a seated experience. To that end, translation was enacted by using the left thumbstick, while a 45° rotation was performed by moving the right thumbstick to the right and left. When initiating movement, the user would instantly reach max speed and likewise stop immediately, avoiding inducing cybersickness. Real-walking was also possible for traversing the VE.

2.2.2 Design of the Avatar

The representation of the player's body can influence the VRMI interaction [24], thus the design of the avatar was discussed with respect to the autistic perception. Some studies have shown improvements in social skills when

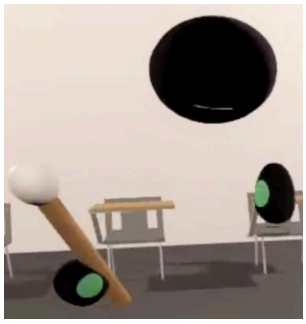


Figure 2. *The default Normcore avatar used to represent the users in the VE.*

using a highly realistic-looking avatar, especially regarding the use of realistic facial expressions [25]. Yet, others find that the individual can struggle when too many stimuli are present, i.e., to recognize and distinguish facial expressions from one another [26]. Moreover, when asked about the avatar design, the therapists were concerned about using a realistic avatar, being potentially intimidating. Because of the conflicting views and with the feedback from the therapists, we decided to utilise the default Normcore avatar, which is a simple iconic-looking avatar (Figure 2), to keep with our user-centered approach. The avatar had a black sphere for the head with a white, smiling mouth. The mouth changed size based on the microphone input. Remotely, the users would see other users' hands as simple mittens to reduce the amount of data needed to be sent and thus the overall latency, which also minimises the risk of cybersickness [24]. Locally, the mittens were exchanged with two five-fingered hand models, created in Autodesk Maya. This change drew upon the VRMI interaction [24] and also aimed at adding avatar-realism, to induce a higher level of perceived presence [27], and thus to increase the ecological validity of the environment that is of importance for social neuroscience.

2.2.3 Implementing the First Iteration

The application was developed in Unity 2019.4.19f1, with the SteamVR plug-in and the multiplayer framework Normcore. With Normcore, the therapist and their client can connect to the same room, which is defined by an app key. For voice communication, Normcore uses a VoIP (voice over internet protocol) that is optimized for minimal latency. This feature can for instance allow the music therapist to also sing with the child during the sessions.

3. EVALUATION BY A MUSIC THERAPIST

This evaluation had two objectives: gathering design insights to improve the application, and validating the potential use of the application in the music therapist's sessions. The following subsections first present the method used, then the experiment's findings, and finally detail the respective design adjustments that were made.

3.1 Method for the Evaluation

A remote VR experiment was conducted with a music therapist from the preliminary interview. Without the opportunity to provide the music therapist with an HMD, the experiment was conducted via videocall, where a researcher shared their view from VR. The therapist was asked to 'puppeteer' the researcher to interact within the VE, and to vocalise her thought about what she saw, in line with the thinking aloud method [28]. The experiment lasted about an hour. Likewise the preliminary interview, the researchers inquired about the design, the interaction with the instruments, and the possibilities for training social and communication skills. Data were analysed with traditional coding; the feedback from the therapist was transcribed and categorised into emerging themes by three researchers [29]. The categories were compared and discussed between the researchers which led to the final categories, from where data was interpreted.

3.2 Findings from the Experiment

Three themes emerged from the data, namely, usability, social interaction, and functionality, as well as seven categories (Table 1). They are described hereafter.

THEMES	CATEGORIES
Usability	• Ease of use • Versatility • Familiarity
Social interaction	• User expression • Clear participation
Functionality	• Instruments • Environmental purpose

Table 1. *Overview of the emerging themes with their definitions and related categories from the experiment with the music therapist.*

Beginning with **usability**, the therapist found the instruments intuitive to interact with and familiar, which could be perceived as reassuring and engaging. Indeed, the xylophone was easily recognisable, and for the Looper, she said "*...even if I had never seen it before, there is still something intuitive and familiar, about this table with this ball that is going around. It kind of reminds [me] of air hockey or something similar. You get an urge to approach [it] and do something with it*". Furthermore, she highlighted the positive effects of the Looper's gaming-like features. Yet, concern about its versatility arose due to the idiosyncratic preferences and needs of autistic individuals, making it potentially too advanced for some children.

In terms of **social interaction**, what contributions each user had made was unclear, possibly hindering the perceived sense of agency. To that respect, displaying more easily distinguishable sounds was advised to better hear one's contributions. Though, she highlighted the strong social potential of the application. For instance, a simple gesture such as asking the other player for a cube that

could be of reach could constitute a great exercise for training social interaction skills. At last, the therapist found the avatar design too simplistic, as only the mouth was reacting to the microphone input, which could be perceived as unsafe. Indeed, facial expressions are important during non-VR therapists' sessions to adapt their behaviour according to the child's state. However, she added still being able to read their client's body language, which is another viable method for understanding their mood, and to adjust the soundscape accordingly.

Regarding **functionality** she warned about potential unnecessary distractions. She specifically mentioned that stacking the cubes could take over the child's attention and be perceived as the main task, thus removing the focus from creating sounds together. Moreover, the colourful items and furniture could also grab the child's attention. Hence, the therapist proposed two solutions: using more neutral colours for the furniture or making the salient items into VRMIs. Regarding this second possibility, the lockers could produce a sound whenever opened, and musical drawings could be created by drawing on the blackboard. Furthermore, the therapist suggested to exploiting more the virtuality of the xylophone, by adding visual feedback or switching colours in line with the different soundmodes of the cubes. At last, regarding the Looper, more soundmodes were desired, as well as simplifying the backbeat and its relation to the white dots on the track more obvious.

The therapist concluded that the application would be used in their sessions with the mentioned adjustments. "*There are some elements that will be lost [in VR], but what other things can we then add? It is the gestures and intuition and mood that we need to compromise because there is something other, that [VR] can do, which is interesting. And it will be exciting to find out what that is. So, it can be an extension of what you otherwise can do in the musical room*". She added being excited to participate in future iterations and tests to discover how to properly use it and maybe to expand the application even more.

3.3 Redesigning the Instrument

Based on the therapist' feedback it was decided to enhance the Looper and remove the xylophone, which was considered unnecessary. To that end, brainstorming sessions were conducted between the authors. This process led to replacing the Looper's track with a railroad, as autistic children often enjoy trains [30]. Hence, a locomotive was used instead of a ball to loop over rails. Gates replaced the cubes to remove attention from stacking the objects. The gates could be placed across the rails so that the locomotive could drive through and activate the corresponding sound and additional visual effect (music notes jumping out from the gate). Authors hypothesised that these changes would make the Looper more usable and discoverable - and thus more adapted to the target group previously described by the therapist in section 3.

In addition to these changes, interaction possibilities were added to provide more control over the created soundscape. The controls are summarised in Table 2. Controllers were assembled in a control panel (see Figure 4). More distinguishable sound modes were also implemented, i.e., marimba, drums, 'magic' sounds (whistle, plings, and whooshes). The VE resulting from this redesign is displayed in Figure 3.

Controller	**Purpose**	**Other feedback**
Lever	Train drives faster/slower.	-
+/- buttons	Volume of the gate sounds is decreased/increased.	Transparency of the smoke is decreased/increased.
Rotator	Pitch of the gate sounds is decreased/increased.	Size of the smoke is increased/decreased.
Whistle	Whistles and the train makes a horn sound.	Smoke colour changes to one of the avatar colours.

Table 2. *Controllers' functions and related feedback.*

4. CONTINUED DESIGN PROCESS WITH TWO PSYCHOLOGISTS

A second design iteration was conducted with two psychologists with two objectives: refining the design according to their specific needs and validating the potential use of this application in their sessions. This section will first present the method that was used, and then the design insights that emerged and led to several design adjustments.

4.1 Method for the Second Design Iteration

The psychologists are autism experts working in a day hospital and use digital tools in their daily interventions (i.e., video games with a projected screen, tablet, and robots). Also, they have previously worked with third author, and thus have experience with evaluating the use of augmented reality technologies in sessions with autistic children [31]. With their expertise and cooperation, future testing with the target group was planned (more about this in section 6.2).

For iterating the VR application, the psychologists participated in three online meetings, where the first author would share their screen to showcase the VR application.

4.2 Design Insights from the Psychologists

Similar to the music therapist's feedback (in section 3), notes were made about the avatar design and the ability to better see each others' contributions. Additional comments were made about the design of the VE, potential interaction and locomotion difficulties, audio effects, as well as the training and introduction to the system for the children. Their different insights are described hereafter.

4.2.1 The Avatar Design

The psychologists suggested making the avatar more expressive. Hence, we added eyes and a nose and made it

Figure 3. *The VE and the interaction stations. Top left to right: the gates, the avatar, and colour buttons. Bottom: the railroad system with control panel on the side.*

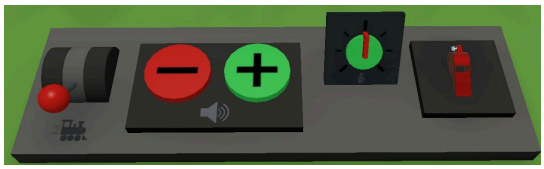


Figure 4. *The control panel for the railroad system. From left to right; lever, +/- buttons, rotator, and whistle.*

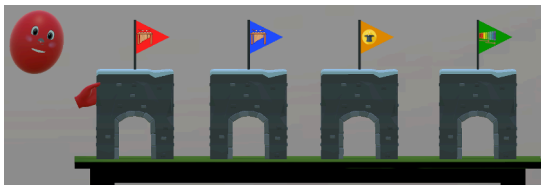


Figure 5. *The flags on the gates change colours depending on the colour of the player who last interacted with it. Also, to the left; the revised avatar design in red.*

possible to change the avatar colour by pressing one of the colour buttons placed on the wall. The colour allowed to better visualise what contributions each user made. When interacting with one of the gates, the flag on top of the gate changed colour corresponding to the interactor's avatar colour (Figure 5).

4.2.2 Adaptation to Autism Perception

To adapt to the specific autism perception, the psychologists asked to simplify the space and to avoid unnecessary distractions, similar to the music therapists. Thus, only objects meant for interaction were kept (the colour buttons, the gates, the railroad and its control panel). Then, as ASD can lead to mobility issues, the interactions were further discussed. To minimise errors while operating the instrument, big buttons and handles were suggested for the control panel. Also, the gates were given a relatively big size and mass, making sure that they would not easily tip over. Psychologists also asked that the session could be performed from a seated position. At last, paying attention to sound modes that the gates are given was deemed important, as people on the autism spectrum typically have difficulties processing sensory inputs [1]. It was then advised to avoid high-pitched sounds, resulting in replacing some of the sounds with low-pitched instruments (e.g., marimba, drums, and rhodes).

4.2.3 Virtuality

The psychologists stressed the importance of using salient elements of virtuality, to help the children to separate the virtual from the real world. Here, the non-realistic avatar and visual effects could serve as useful signals.

4.2.4 Other

The psychologists suggested adding items that the children could break or throw around, as this is common in their experience and could be used for communicating. This was noted as future work. Tutorials for the VR tool were also suggested; however, it was later mentioned to be unnecessary as the therapist could guide the child in the VE.

4.2.5 Implementation of Second Iteration

The application was developed in Unity 2020.3.18f1 for the Oculus Quest 2. The framework used for setting up VR interaction was XR Interaction Toolkit, and Normcore was used to provide multiplayer functionality.

5. PRE-TESTS WITH NON-AUTISTIC CHILDREN

Pre-tests were conducted with non-autistic children to assess their acceptability, usability, and perception of the VR application, We considered that gathering positive feedback about the acceptability and usability of the application with non-autistic children could validate both of the future testing with the psychologists.

5.1 Method for the Pre-tests

A parent from author's professional network invited his child's class to visit the *Multisensory Experience Lab* at Aalborg University in Copenhagen. In connection with the visit, the 25 children (age 9) were recruited as proxy participants for evaluating the VR artefact. The children came

in groups of 4 or 5, accompanied by their respective teachers. Each child was given about 3 minutes to explore the environment alone. They were aided to wear the HMD and asked if the image that they saw was blurred. Based on their answer, the HMD was further adjusted until the child agreed. Children were provided with guidance if they asked questions or stood still. The view from the HMD was streamed to a PC monitor so that the researcher could observe the child's behaviour in relation to their actions in VR.

During the testing, the first author took notes of the child's behaviour, interaction (ease of use), and motivation, both in real life with the HMD, and by looking at the screen. Comments from the other children were also taken into account. As the pre-test mainly aimed at evaluating the acceptability and usability when wearing the HMD, the bias created by the other children observing and commenting on the child testing the environment was considered as not having a non-significant impact.

5.2 Findings from the Observations

Findings suggested that more guidance was needed. This could be done by having a guide with them in VR or including tutorials for the different features.

The need for small adjustments to the control panel appeared; a heavier hinge joint for the lever would decrease the sensitivity of the tempo of the locomotive, and greater rotational sensitivity for the rotator, to decrease the risk of straining the wrist. The sizing of the furniture also had to be decreased, as many children found it difficult to reach some of the items on the tables. Also, to promote cooperation between the users, the size of the railroad had to be reduced so that the ends of the table could be closer. This feature has to be included for future testing with autistic children to prompt the social VR experience.

Some children tried methods to 'break the system', which aligns with psychologists' views about how autistic children would interact with the application. For instance, children threw the gates around the virtual room, which seemed enjoyable to them. Others tried to exit the room, which at times resulted in the virtual player clipping through the walls of the room and in an endless fall. This did not scare the children. It was clear that prevention of clipping through the virtual walls, as well as a reset or 'clean up' button, were needed. Indeed, the alternative would be to terminate the VR experience and restart the app, which could break the flow in the session.

On the technical side, the Oculus Quest 2 did not seem too heavy or uncomfortable for the children. The battery lasted for about 2 hours, which is sufficient for one or two full sessions of MT.

6. DISCUSSION AND FUTURE WORKS

The VR-based MT application presented in this paper is based on feedback from professionals experienced in working with children and adolescents diagnosed with ASD, thereby creating the foundation for future studies investigating its efficiency to increase the social capabilities of the target group. The application allows a child on the autism spectrum to log into an online VE without physically interacting with a music therapist, thereby reducing the risk for social anxiety. Additionally, VR enables the music therapist to access virtual instruments that are not accessible in real life and are potentially easy for an autistic child to interact with.

6.1 Adapting to the Autistic Perception

Both the music therapist and the psychologists encouraged to simplify the VE to adapt to the autism perception. More specifically, only meaningful virtual objects should be kept visible to not potentially catch the child's attention, removing focus from the social interaction.

According to the participating experts, the designed Looper instrument could potentially offer sufficient usability for an autistic child to use - with certain adjustments. Clarifying the perception of what contributions each user had made, was deemed fundamental to facilitate social interaction features (e.g., joint attention). More distinguishable sounds were suggested by the music therapist to feature as the separator of user contributions. This resulted in marimba, drums, and 'magic' sounds (whistle, plings, and whooshes) sounds. In contrast, the psychologists requested low-pitched sounds as these are more easily processed by individuals on the autism spectrum, which were, therefore, implemented for the second iteration. In future work, the sounds could be customised to their preferences and need to allow for a more versatile tool.

Concerning the avatar, all the therapists advocated for using a simplified model rather than realistic. Though, findings from a recent VR study conducted with 7 children with mild autism show improvements in social skills when using highly realistic-looking avatars [25]. We can hypothesize that this contrast may be due to the fact that the therapists in this paper commonly work with children with severe to moderate autism, contrary to this study's participants. This discrepancy calls for further research to evaluate if therapists' needs regarding the avatar design are correlated to autism severity.

It should be noted that there are limitations to the VR system. As mentioned by both music therapists and psychologists, facial expressions can be an important aspect of recognising the child's emotions. With body language being the alternative, as noted by the music therapist, two types of avatars could convey more expressive movements: an avatar with inverse kinematics - and a full-body avatar with additional equipment such as sensors attached to hips and feet. However, with the design consideration of using a stand-alone technology, adding equipment could be an issue, as well as possibly hinder the child's acceptance.

As the psychologists had suggested, it was found that some non-autistic children who tested the application would throw around with the virtual objects or 'break the system' by moving outside the room's boundaries. Such behaviours could indicate a strong excitement or a will to examine the limits of the system. Moreover, the psychologists noted that trying to break the system could be understood in communicative ways and could be beneficial

to incorporate.

6.2 Limitations and Future Works

With the COVID-19 regulations that were enforced during the design process, neither the music therapists nor psychologists could try the VR application for themselves. Thus, additional evaluations must be conducted with the autism experts in VR before an evaluation can be conducted with autistic children.

In a previous augmented reality study [31], before experiencing the VE, the practitioners were worried of feeling enclosed within the HMD and that it would not be adapted for autistic children. However, neither the practitioners or non-autistic children mentioned this in the present study.

It should be mentioned that the application is developed by and in co-operation with non-autistic people. Although the pre-test with school children provided valuable insights, it only considered the perception of non-autistic users. Thus, the VE has not yet been tested with autistic participants and cannot account for potential hypersensitivity that some autistic children display. Especially, the choice of the audiovisual effects generated by an activated gate may receive negative feedback from children on the autism spectrum. While we do not question that practitioners could teach the children how to interact in VR, conducting tutorials with both practitioners and children could help to minimize potential risks due to hypersensitivity. Indeed, it could gradually allow them to get used to the VE or to individualize the VE based on their perception.

Previous work show that VR can prompt social interaction [7–10], yet question remains about the ecological validity of VR interventions. As MT is time consuming and thus require months or years before noticeable results are showing [7, 23], longitudinal studies are necessary to evaluate this question.

This paper focused on designing a multiplayer VR tool to meet the needs of some primary users, i.e., music therapists and psychologists. Future focus will be on whether they can promote social and communications skills by adding the tool in common autism interventions. To that end, the VR application will be evaluated at the day hospital in collaboration with the two psychologists, with six children (+11 years) with severe to moderate ASD. The children will do a 20-minute session once a week over several weeks to assess their acceptability, usability, and the evolution of the social interaction.

Acknowledgments

We would like to thank Professor in Music Therapy Ulla Holck and Music Therapist Anne Cathrine Hulthin Andersen for the great guidance, that shaped this study. We are grateful for the advice and feedback provided by psychologist and researcher Olivier Duris, clinical psychologist and Ph. D. student Charlotte Labossière, and psychiatrist and physician director Marie-Noëlle Clément at the Association CEREP-PHYMENTIN, Day hospital André Boulloche.

Funding Information: The participation to the conference was supported by the European Art Science and Technology Network (EASTN-DC).

7. REFERENCES

[1] C. Lord, T. S. Brugha, T. Charman, J. Cusack, G. Dumas, T. Frazier, E. J. H. Jones, R. M. Jones, A. Pickles, M. W. State, J. L. Taylor, and J. Veenstra-VanderWeele, "Autism spectrum disorder," *Nature Reviews Disease Primers*, vol. 6, no. 1, p. 5, Jan. 2020.

[2] S. Bellini, "The development of social anxiety in adolescents with autism spectrum disorders," *Focus on autism and other developmental disabilities*, vol. 21, no. 3, pp. 138–145, 2006.

[3] B. Chamberlain, C. Kasari, and E. Rotheram-Fuller, "Involvement or isolation? The social networks of children with autism in regular classrooms," *Journal of autism and developmental disorders*, vol. 37, no. 2, pp. 230–242, 2007.

[4] M. Ghaziuddin, N. Ghaziuddin, and J. Greden, "Depression in persons with autism: Implications for research and clinical care," *Journal of autism and developmental disorders*, vol. 32, no. 4, pp. 299–306, 2002.

[5] K. Bottema-Beutel, S. K. Kapp, J. N. Lester, N. J. Sasson, and B. N. Hand., "Avoiding ableist language: Suggestions for autism researchers," *Autism in Adulthood*, pp. 18–29, 03 2021.

[6] L. Kenny, C. Hattersley, B. Molins, C. Buckley, C. Povey, and E. Pellicano, "Which terms should be used to describe autism? perspectives from the uk autism community," *Autism*, vol. 20, no. 4, pp. 442–462, 2016, pMID: 26134030. [Online]. Available: https://doi.org/10.1177/1362361315588200

[7] D. G. Provenzano and A. Stannard, "Music therapy for communication in children with autism spectrum disorder," Academic Festival, Event 7. Digital Commons, 2020.

[8] A. Adjorlu, N. B. B. Barriga, and S. Serafin, "Virtual reality music intervention to reduce social anxiety in adolescents diagnosed with autism spectrum disorder," in *16th Sound and music computing conference*. Sound and Music Computing Network, 2019, pp. 261–268.

[9] E. Gal, P. L. T. Weiss, and M. Zancanaro, *Using Innovative Technologies as Therapeutic and Educational Tools for Children with Autism Spectrum Disorder*. New York, NY: Springer New York, 2019, pp. 227–246.

[10] M. Geretsegger, C. Elefant, K. A. Mössler, and C. Gold, "Music therapy for people with autism spectrum disorder," *Cochrane Database of Systematic Reviews*, no. 6, 2014.

[11] A. Adjorlu, "Virtual reality therapy," in *Encyclopedia of Computer Graphics and Games (ECGG)*. Springer, 2018.

[12] S. Burke, T. Bresnahan, T. Li, K. Epnere, A. Rizzo, M. Partin, R. Ahlness, and M. Trimmer, "Using virtual interactive training agents (ViTA) with adults with autism and other developmental disabilities," *Journal of Autism and Developmental Disorders*, vol. 48, 03 2018.

[13] H. Ip, S. Wong, D. Chan, J. Byrne, R. Li, V. Yuan, K. Lau, and W. Y. Joe, "Virtual reality enabled training for social adaptation in inclusive education settings for school-aged children with autism spectrum disorder (ASD)," vol. 9757, 07 2016, pp. 94–102.

[14] S. Parsons and P. Mitchell, "The potential of virtual reality in social skills training for people with autistic spectrum disorders," *Journal of intellectual disability research*, vol. 46, no. 5, pp. 430–443, 2002.

[15] D. Johnston, H. Egermann, and G. Kearney, "Innovative computer technology in music based interventions for individuals with autism - moving beyond traditional interactive music therapy techniques," *Cogent Psychology*, vol. 5, 11 2018.

[16] M. Shahab, A. Taheri, M. Mokhtari, A. Shariati, R. Heidari, A. Meghdari, and M. Alemi, "Utilizing social virtual reality robot (V2R) for music education to children with high-functioning autism," *Education and Information Technologies*, Jan. 2021.

[17] L. Bryce, M. Sandler, L. Koreska Andersen, A. Adjorlu, and S. Serafin, "The Sense of Auditory Presence in a Choir for Virtual Reality," in *Audio Engineering Society Convention 149*. Audio Engineering Society, Oct. 2020.

[18] A. Brungardt, A. Wibben, A. F. Tompkins, P. Shanbhag, H. Coats, A. B. LaGasse, D. Boeldt, J. Youngwerth, J. S. Kutner, and H. D. Lum, "Virtual reality-based music therapy in palliative care: A pilot implementation trial," *Journal of Palliative Medicine*, vol. 24, no. 5, pp. 736–742, 2021.

[19] S. Parsons, N. Yuill, J. Good, and M. Brosnan, "'Whose agenda? Who knows best? Whose voice?' Co-creating a technology research roadmap with autism stakeholders," *Disability & Society*, vol. 35, no. 2, pp. 201–234, 2020.

[20] J. Preece, H. Sharp, and Y. Rogers, *Interaction Design: Beyond Human-Computer Interaction*, 4th ed. Wiley, 2015, ch. 9.2.

[21] L. B. Baltaxe-Admony, T. Hope, K. Watanabe, M. Teodorescu, S. Kurniawan, and T. Nishimura, "Exploring the creation of useful interfaces for music therapists," in *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, ser. AM'18. New York, NY, USA: Association for Computing Machinery, 2018.

[22] M. Geretsegger, U. Holck, J. A. Carpente, C. Elefant, J. Kim, and C. Gold, "Common characteristics of improvisational approaches in music therapy for children with autism spectrum disorder: Developing treatment guidelines," *Journal of music therapy*, vol. 52, no. 2, pp. 258–281, 2015.

[23] U. Holck, "Turn-taking in music therapy with children with communication disorders," *British Journal of Music Therapy*, vol. 18, no. 2, pp. 45–54, 2004.

[24] S. Serafin, C. Erkut, J. Kojs, N. C. Nilsson, and R. Nordahl, "Virtual reality musical instruments: State of the art, design principles, and future directions," *Computer Music Journal*, vol. 40, pp. 22–40, 2016.

[25] J. F. Herrero and G. Lorenzo, "An immersive virtual reality educational intervention on people with autism spectrum disorders (ASD) for the development of communication skills and problem solving," *Education and Information Technologies*, vol. 25, p. 1689–1722, 05 2020.

[26] C. Keating and J. Cook, "Facial expression production and recognition in autism spectrum disorders," *Child and Adolescent Psychiatric Clinics of North America*, vol. 29, 04 2020.

[27] D. Roth, J.-L. Lugrin, D. Galakhov, A. Hofmann, G. Bente, M. Latoschik, and A. Fuhrmann, "Avatar realism and social interaction quality in virtual reality," in *2016 IEEE Virtual Reality (VR)*. IEEE, 03 2016, pp. 277–278.

[28] K. A. Ericsson and H. A. Simon, *Protocol Analysis: Verbal Reports as Data*. The MIT Press, 04 1993, (accessed: 14.04.2022). [Online]. Available: https://doi.org/10.7551/mitpress/5657.001.0001

[29] T. Bjørner, *Qualitative Methods for Consumer Research: The Value of the Qualitative Approach in Theory and Practice*. Hans Reitzels Forlag, 2015, ch. 3-4.

[30] N. E. Scheerer, E. Birmingham, T. Q. Boucher, and G. Iarocci, "Attention capture by trains and faces in children with and without autism spectrum disorder," *PLOS ONE*, vol. 16, no. 6, pp. 1–28, 06 2021.

[31] V. Bauer, T. Bouchara, and P. Bourdot, "Designing an Extended Reality Application to Expand Clinic-Based Sensory Strategies for Autistic Children Requiring Substantial Support: Participation of Practitioners," *IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, vol. 1, pp. 254–259, 2021.

Appendices

- **Appendix A:** First iteration test: `https://youtu.be/J1LRAL_cARs`
- **Appendix B:** Second iteration test: `https://youtu.be/NCp9D0dHxt0`

BARWISE COMPRESSION SCHEMES FOR AUDIO-BASED MUSIC STRUCTURE ANALYSIS

Axel Marmoret **Frédéric Bimbot**
Univ Rennes, Inria, CNRS, IRISA, France.
firstname.name@irisa.fr

Jérémy E. Cohen
Univ Lyon, INSA-Lyon, UCBL,
UJM-Saint Etienne, CNRS, Inserm,
CREATIS UMR5220, U1206, France
jeremy.cohen@cnrs.fr

ABSTRACT

Music Structure Analysis (MSA) consists in segmenting a music piece in several distinct sections. We approach MSA within a compression framework, under the hypothesis that the structure is more easily revealed by a simplified representation of the original content of the song. More specifically, under the hypothesis that MSA is correlated with similarities occurring at the bar scale, this article introduces the use of linear and non-linear compression schemes on barwise audio signals. Compressed representations capture the most salient components of the different bars in the song and are then used to infer the song structure using a dynamic programming algorithm. This work explores both low-rank approximation models such as Principal Component Analysis or Nonnegative Matrix Factorization and "piece-specific" Auto-Encoding Neural Networks, with the objective to learn latent representations specific to a given song. Such approaches do not rely on supervision nor annotations, which are well-known to be tedious to collect and possibly ambiguous in MSA description. In our experiments, several unsupervised compression schemes achieve a level of performance comparable to that of state-of-the-art supervised methods (for 3s tolerance) on the RWC-Pop dataset, showcasing the importance of the barwise compression processing for MSA.

1. INTRODUCTION

Music Structure Analysis (MSA) consists in subdividing a music piece in several distinct parts, which represent a mid-level description of a song. Structure is an important part of music, and could be the main difference between music and random noise [1]. In that sense, MSA has been extensively studied in Music Information Retrieval (MIR), see [1, 2] for overviews. Practically, in the era of computational analysis of music with Machine Learning algorithms, structure could be an important mid-level feature for tasks such as cover detection, genre recognition, music summarization, for better recommendation systems, and many other tasks. Segmentation is usually based on criteria such as homogeneity, novelty, repetition and regularity [2]. When performed algorithmically, MSA often relies on similarity criteria within passages of a song summarized in an autosimilarity matrix [3–11], in which each coefficient represents an estimation of the similarity between two musical fragments. Related work mainly splits in two categories: blind (unsupervised) and learning-based (supervised) techniques. Blind segmentation techniques, such as this work, do not use training datasets.

Similarity between two frames can be obtained from the feature representation of the signal, such as the Short-Time Fourier Transform (STFT) of the song [3]. Boundaries can then be detected as points of strong dissimilarity between the near past and future of a given instant, as in [3]. Still, recent works have aimed at designing new representations of the original music, able to capture the similarity between two frames while maintaining a high level of dissimilarity between dissimilar frames, either in a blind [2, 4–9] or in a supervised [10, 11] fashion. This generally consists in projecting the data in a new feature space and computing the similarity in the feature space. In particular, McFee and Ellis made use of spectral clustering as a segmentation method by interpreting the autosimilarity as a graph and clustering principally connected vertices as segments [5]. This work performed best among blind segmentation techniques in the last structural segmentation MIREX campaign in 2016 [12]. Recently, we used tensor decomposition (Nonnegative Tucker Decomposition, NTD) as a way to describe music as barwise patterns, which then served as features for the computation of the autosimilarity matrix [9].

In most former works, the similarity is expressed between beatwise aligned features, as in [5, 6]. Instead, the present work considers that repetitions are more prone to happen at the barscale, and hence focuses on barwise aligned features, as in [9]. A comparison between both beatwise and barwise aligned features has been made in [10], without one singling out. In this work, we first explore low-rank approximation methods to perform barwise dimensionality reduction using Principal Component Analysis (PCA) and Nonnegative Matrix Factorization (NMF), and then design "piece-specific" Auto-Encoding Neural Networks (AE). While PCA and NMF are standard methods nowadays, the work presented in this article is the first one, as far as we know, to consider them for barwise dimensionality reduction.

Copyright: © 2022 Axel Marmoret et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

This work extends the concept of barwise compression by introducing a song-dependent autoencoder, *i.e.* an AE which is specifically trained to compress a given song. This technique is called "Single-Song Auto-Encoding" and the latent representation of each bar in the AE is used as a sequence of features for segmenting the song. Indeed, recently, Deep Neural Networks (DNN) have lead to some high level of excitement in MIR research, and notably in MSA [8, 13]. In general, DNN approaches rely on large databases which make it possible to learn a large number of parameters, which in turn yields better performance than previously established machine learning approaches. In particular, to the best of the authors' knowledge, the current state-of-the-art approach for the structural segmentation of Popular music is a supervised CNN developed by Grill and Schlüter [13]. This is the consequence of the ability of DNNs to learn complex nonlinear mappings through which musical objects can be expected to be better separated [14]. Hence, while DNNs generally learn "deep" features stemming from multiple examples in a training phase, and then evaluate the potential of learned features [15], our single-song AE approach focuses on the different events within a single song and tries to learn nonlinear latent representations, used to infer the structure.

To study the relevance of the barwise compression approach, this work evaluates different dimensionality reduction techniques on the RWC-Pop dataset [16] in their audio form using various time-frequency features. Segmentation results on this database show levels of performance which are outperforming the current blind state-of-the-art [3,5,9], and our best performing model outperforms the supervised state-of-the-art [13] with 3-seconds tolerance.

The rest of the article is structured as follows: Section 2 details the motivations and approaches for barwise music compression. Section 3 presents the various compression schemes used in this work. Section 4 presents the segmentation process. Section 5 reports on the experimental results.

2. BARWISE MUSIC COMPRESSION

2.1 Motivations

The underlying idea of this work is that music structure can be related to compression of information. Indeed, a common view of music structure is to consider structural segments as internally coherent passages, and automatic retrieval techniques generally focus on finding them by maximizing homogeneity and repetition and/or by setting boundaries between dissimilar segments, as points of high novelty [2]. In the context of compressed representations, each passage is transformed in a vector of small dimension, compelling this representation to summarize the original content. From this angle, similar passages are expected to be represented by similar representations, as they share underlying properties (such as coherence and redundancy), while dissimilar passages are bound to create strong discrepancies at their boundaries. Thus, we expect that compressed representations will enhance the original structure while reducing incidental signal-wise properties which do not contribute to the structure. To the best of our knowledge, this work, our former work [9], and recent work by Wang *et al.* [10] are the only barwise compression schemes with application to structural segmentation of music.

Conversely to fixed-size frame analysis, barwise computation guarantees that the information contained in each frame does not depend on the tempo, but on the metrical positions, which is a more abstract musical notion to describe time. As a consequence, comparing bars is more reliable than comparing frames of arbitrarily fixed size as it allows to cope with small variations of tempo. In addition, pop music (*i.e.* our case study) is generally quite regular at the bar level: repetitions occur at the bar scale and motivic patterns tend to develop within a limited number of bars, suggesting that frontiers mainly sit between bars. To support this claim, Table 1 of Section 5 presents segmentation results when realigning the annotation on the closest downbeat.

Accordingly, the proposed method relies on a consistent bar division of music. It also requires a powerful tool to detect bars, as otherwise errors could propagate and affect the performance. Results reported in [9] and in Table 1 tend to show that the madmom toolbox [17] is efficient in that respect on the RWC-Pop dataset. Nonetheless, barwise processing may hinder the retrieval of boundaries delimiting changes of tempo, as discussed in [18]. It also relies on some consistent bar division of music, which is generally the case for contemporary western music, but is not a universal rule, and/or may turn out to be ambiguous.

2.2 Barwise Music Processing

Following our former work in [9], we process music as barwise spectrograms, with a fixed number of frames per bar. Practically, spectrograms are computed using librosa [19] with a low hop length of 32 frames at a sampling rate of 44.1kHz, and downbeats are estimated with the madmom toolbox [17]. This allows us to split the original spectrogram in b barwise spectrograms (b being the number of bars in this song) each containing n_b frames. As bars can be of different lengths (because of tempi differences or small inconsistencies/imperfections in the performance), different bars can contain different numbers n_b of frames. Thus, we define an integer s, the subdivision parameter, which is the desired number of frames in each bar.

Starting from the subdivision s, and from indexes f_s and f_e, respectively the indexes of the closest frames to the downbeats starting and ending the bar, we select all frames $\left\{\frac{f_s+k\times(f_e-f_s)}{s}, 0 \leq k < s, k \in \mathbb{N}\right\}$, *i.e.* equally-spaced frames in the bar to fit the chosen subdivision. This indeed results in barwise spectrograms containing exactly s frames. Other techniques could be applied to reduce the number of frames (for example averaging the content of several frames instead of choosing one), but we did not pursue that lead. We chose $s = 96$ as in [9].

This technique result in b barwise Time-Frequency spectrograms, of size $f \times s$, with f the number of components in the chosen feature, as presented in section 2.3. Compression is generally performed on matrices (as PCA and NMF for instance). In that sense, by vectorizing each of

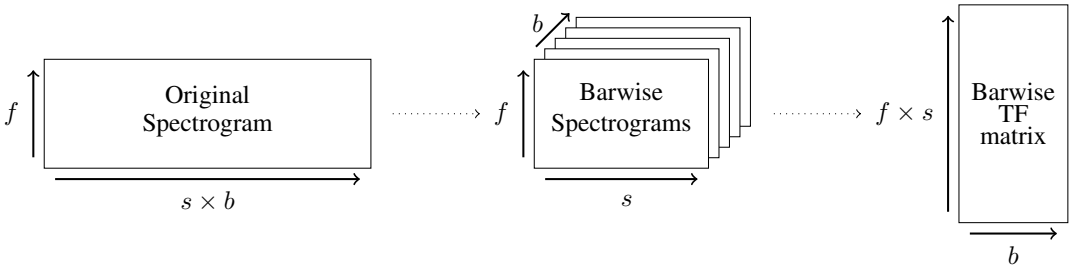


Figure 1: Barwise processing of an input spectrogram, resulting in a barwise TF matrix.

the previous barwise spectrograms, we introduce the "barwise TF" representation, consisting in a matrix of size $b \times (f \times s)$. The aforementioned process is described in Figure 1. Note that we chose to vectorize the time-frequency features, thus discarding the dependencies between these two dimensions.

2.3 Features

Music is represented with different features throughout our experiments, focusing on different aspects of music such as harmony or timbre. These features are presented hereafter.

Chromagram A chromagram represents the time-frequency aspect of music as sequences of 12-row vectors, corresponding to the 12 semi-tones of the classical western music chromatic scale (C, C#, ..., B), which is largely used in Pop music. Hence, $f = 12$, and each row represents the weight of a semi-tone (and its octave counterparts) at a particular instant.

Mel spectrogram A Mel spectrogram corresponds to the STFT representation of a song, whose frequency bins are recast in the Mel scale. These bands account for the exponential spread of frequencies throughout the octaves. Mel spectrograms are dimensioned following the work of [13] (80 filters, from 80Hz to 16kHz), hence $f = 80$. STFT are computed as power spectrograms.

Log Mel spectrogram (LMS) A Log Mel spectrogram (abbreviated "LMS") corresponds to the logarithmic values of the precedent spectrogram, hence f is still equal to 80. This representation accounts for the exponential decay of power with frequency observed in STFT.

Nonnegative Log Mel spectrogram (NNLMS) Nonnegative Log Mel spectrogram (abbreviated "NNLMS") is a nonnegative version of the precedent LMS. This representation is motivated by the fact that some compression algorithms are nonnegative (in particular, NMF), and thus need nonnegative inputs. NNLMS are computed as $\log(Mel + 1)$ where Mel represent the coefficients of the Mel spectrogram, which are nonnegative, and $\log$ the elementwise logarithm.

MFCC spectrogram Mel-Frequency Cepstral Coefficients (MFCCs) are timbre-related coefficients, obtained by a discrete cosine transform of a Log Mel spectrogram [1]. This spectrogram contains $f = 32$ coefficients, following [20].

[1] Note that we use the default librosa settings for MFCC, hence the Log Mel spectrogram used for MFCC is not the same as for the LMS.

3. COMPRESSION SCHEMES

3.1 Low-Rank Approximations

Given a collection of input vectors $\{x_i\}_{i=1,\dots,b} \in \mathbb{R}^n$ stored in a matrix X, low-rank approximations search for projections in a low-dimensional subspace which approximates the input vectors. We study two different low-ranks approximations methods, namely PCA and NMF.

PCA is maybe the most well-known and used low-rank approximation technique. It can be seen as a singular value decomposition applied on the debiased matrix $X - \mu$, where μ is the mean of the columns of X. In short, we obtain an approximation $X - \mu \approx WH$ where $W \in \mathbb{R}^{n \times d_c}$ is orthogonal, and $H \in \mathbb{R}^{d_c \times b}$ is the product of a diagonal and an orthogonal matrix. The rank of the approximation d_c is chosen by the user, and should be smaller than both dimensions.

NMF is similar to PCA but does not impose orthogonality, nor does it unbias the data. Instead, it builds a cone containing the data, which extreme rays are the columns of W. To achieve this, nonnegativity is required elementwise on both W and H. The low-rank approximation is in fact the solution of an optimization problem, here:

$$\underset{W \geq 0, H \geq 0}{\arg\min} \|X - WH\|_F^2 \tag{1}$$

using the Frobenius norm as a loss function. NMF typically yields more interpretable features than PCA, but it is much harder to compute, and requires that the data is mostly nonnegative [21]. In both cases, H is the barwise compressed representation.

3.2 Autoencoders

Autoencoders are neural networks, which, by design, perform unsupervised dimensionality reduction. Throughout the years, AE have received increasing interest, notably due to their ability to extract salient latent representations without the need of large amount of annotations. Recently, AE also showed great results as a generation tool [22, 23]. Still, as presented in [23], PCA and AE are competitive as compression schemes, and PCA even leads to lower reconstruction error when AE are too "simple" (in particular when networks are linear or "shallow", *i.e.* with only a few layers).

Practically, given a generic entry $x \in \mathbb{R}^n$, an AE learns a nonlinear function f with parameters θ (weights and biases of the network) such that $\hat{x} = f(x, \theta) \in \mathbb{R}^n$ reconstructs x

as faithfully as possible. This is achieved by minimizing a given loss function such as the Mean Square Error (MSE) between x and $\hat{x}$ w.r.t. parameters θ.

$$\arg\min_{\theta} \text{MSE}(x, \hat{x}) = \frac{1}{n} \sum_{i=0}^{n-1} (x_i - \hat{x}_i)^2 \,. \qquad (2)$$

Other metrics can be used, such as the Kullback-Leibler divergence, but we restrict this work to MSE.

An autoencoder is divided into two parts: an encoder, which compresses the input $x \in \mathbb{R}^n$ into a latent representation $z \in \mathbb{R}^{d_c}$ of smaller dimension (generally, $d_c \ll n$), and a decoder, which reconstructs $\hat{x}$ from z. While Autoencoders generally learn a common latent representation for an entire dataset, our technique optimizes a network for each song. This framework is called "single-song autoencoding".

In this work, layers are of two types: fully-connected, or convolutional. Convolutional layers lead to impressive results in image processing due to their ability to discover local correlations (such as lines or edges), which turn to higher-order features with the depth of the network [24] [25, Ch. 9]. While local correlations are less obvious in spectrogram processing [26], Convolutional Neural Networks (CNN) still perform well in MIR tasks, such as MSA [13].

The tested AE is hence a CNN. We use the nowadays quite standard Rectified Linear Unit (ReLU) function as the activation function, except in the latent layer because it could lead to null latent variables. The encoder is composed of five hidden layers: 2 convolutional/max-pooling blocks, followed by a fully-connected layer, controlling the size d_c of the latent space. Convolutional kernels are of size 3x3, and the pooling is of size 2x2. Convolutional layers define respectively 4 and 16 feature maps.

The decoder is composed of 3 hidden layers: a fully-connected layer (inverse of the previous one) and 2 "transposed convolutional" layers of size 3x3 and stride 2x2. A transposed convolution is similar to the convolution operation taken in the backward pass: an operation which takes one scalar as input and returns several scalars as output [27, Ch. 4]. Hence, it is well suited to inverse the convolution process. This network is represented in Figure 2.

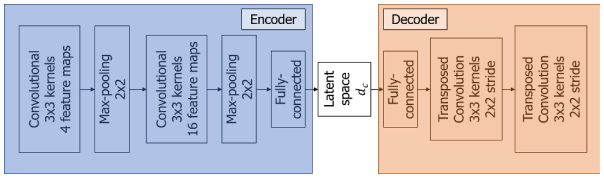


Figure 2: Architecture of the autoencoder

3.3 Dimensions of Compression

Selecting the dimension of the compression d_c, *i.e.* the number of components to use for compressing the input data, is not trivial. In particular, it is well-known that selecting the number of components for NMF is a complex problem, generally leading to manual tuning or dedicated heuristics [21, 28]. We also observed a great influence of the size of the latent space of the AE on the segmentation results, without a clear candidate singling out. In that sense, we will compare values for $d_c \in \{8, 16, 24, 32, 40\}$. Even if such heuristics are standard for PCA (such as the elbow method), as no obvious candidate heuristic stood out relatively to the quality of segmentation, and for fair comparison with other techniques, PCA will be tested with the same d_c values. A clever dimension selection method could be studied in future work.

4. STRUCTURAL SEGMENTATION

4.1 General Principle

The ability of these compression methods to separate and group bars is evaluated on the MSA task, as presented in [29]. Thereby, for a given song in a given representation, we obtain compressed representations z_i for all bars $1 \leq i \leq b$ of the song. These compressed representations are at the heart of the segmentation strategy, via their autosimilarity matrix. Denoting $Z \in \mathbb{R}^{d_c \times b}$ the matrix resulting from the concatenation of all d_c-dimensional z_i barwise compressed representations, its autosimilarity is defined as $Z^T Z$, *i.e.* the $b \times b$ matrix of the dot products between every z_i. These dot products are then normalized to 1, resulting in a matrix of cosine similarities. Examples of autosimilarities are shown in Figure 3.

Segmentation is obtained via the dynamic programming algorithm presented in [9], and inspired from [30]. The principle of dynamic programming is to solve a combinatorial optimization problem by dividing it in several atomic problems, easier to solve, and which solutions can be stitched together to form the global solution. The Viterbi and Dynamic Time Warping (DTW) algorithms are examples of dynamic programming algorithms with a lot of applications in the Audio community. In our context, a "segmentation cost" is computed for every segment in the song. These individual segment costs constitute the atomic problems to solve. Then, the optimal segmentation consists in the global maximum of the sum of the segment costs for all the segments in this segmentation.

The cost of each segment is designed in order to favor their homogeneity. As the autosimilarity matrix represent the similarity between pairs of bars in the song, the higher is this similarity, the most similar are the bars. Hence, the cost of a segment is defined as an aggregated value of the similarity between all pairs of bars in this segment. Practically, this is obtained by convolving a kernel with the autosimilarity restricted to this segment. Hence, to compute the cost of each segment in the song, the algorithm applies a sliding convolving kernel on the diagonal of the autosimilarity matrix. This convolving kernel is a square matrix, the size of which is that of the potential segment.

While sharing the principle of a sliding kernel with the work of Foote [3], largely described in the literature [2], the proposed kernel focuses on homogeneity rather than novelty. When the diagonal of the autosimilarity is structured in several self-similar blocks, the algorithm frames and partitions these blocks.

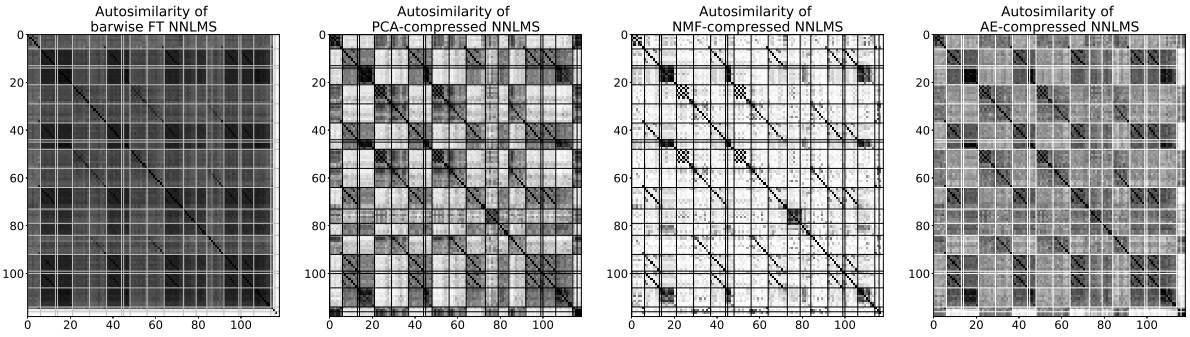


Figure 3: Barwise autosimilarities for the Nonnegative Log Mel spectrogram (left) and of the compressed representation with $d_c = 24$ (from left to right: PCA, NMF and AE) computed on the song "Pop01" from RWC-Pop. Horizontal and vertical lines represent the annotation.

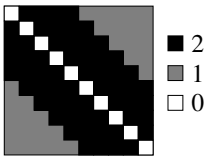


Figure 4: Kernel of size 10

4.2 Technical Details

The design of the kernel defines how to transform bar similarities into segment homogeneity in the form of a cost. A kernel matrix full of ones would imply that the segment cost is the sum of the similarity values in this segment. The proposed kernel is equal to 0 on the diagonal, 2 on the 4 sub and super diagonals, and 1 everywhere else. The diagonal is equal to zero so as to disregard perfect self-similarity of each bar with itself (normalized to 1). The other elements of this kernel are designed so as to emphasize the short-term similarity in the 4 contiguous bars, and still catch longer-term similarity for long segments. It is presented in Figure 4. This kernel performs best in comparative experiments [2] of [9], and, notably, better than a kernel matrix of 1 with a 0 diagonal.

To account for regularity constraints as in [30], the present algorithm adds a regularity penalty $p(n)$ to the plain convolution score, which is a function of the size n (in bars) of the segment. This function $p(n)$ is equal to 0 if $n = 8$, $\frac{1}{4}$ if $n = 4$, $\frac{1}{2}$ if $n \equiv 0 \pmod 2$, and finally 1 otherwise. This penalty function is musically motivated for pop music, as segments are more likely to be of size 4 or 8 bars (especially in RWC-Pop with MIREX10 annotations [30]), than of odd bar size, as shown in Figure 5.

Finally, for all possible segments $[b_1, b_2[$ in the song, the algorithm computes a score:

$$s_{b_1,b_2} = \frac{c_{b_1,b_2}}{c_{k8}^{max}} - p(n), \tag{3}$$

where c_{b_1,b_2} is the convolution cost, and where c_{k8}^{max} is the highest convolution score on this autosimilarity with a kernel of size 8, used for normalization purposes.

[2] https://gitlab.inria.fr/amarmore/musicntd/-/blob/v0.2.0/Notebooks/5%20-%20Different%20kernels.ipynb

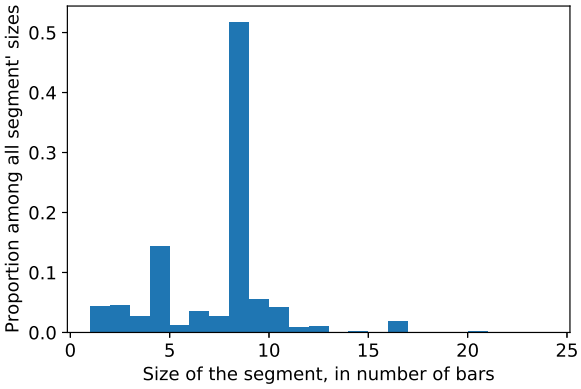


Figure 5: Distribution of segment' sizes in number of bars in MIREX10.

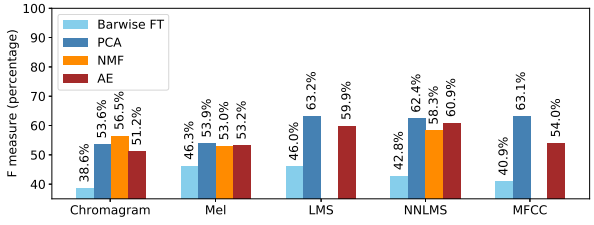
5. EXPERIMENTS

The proposed segmentation pipeline is studied on the RWC Pop dataset, which consists in 100 Pop songs of high recording quality [16], along with the MIREX10 annotations [31]. We compared the barwise compression schemes described in the present work with three blind methods and a supervised method: Foote's novelty kernel [3], Spectral Clustering by McFee and Ellis [5], our former NTD method [9], and the supervised CNN of Grill and Schlüter [13], the latter being the current state-of-the-art. Results for [3] and [5] were computed with the MSAF toolbox [32], and boundaries were aligned to the closest downbeat for fairer comparison (as in [9]). Results for [13] were taken from the 2015 MIREX campaign [12].

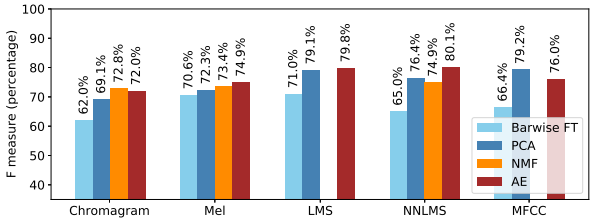
We focus on boundary retrieval and ignore segment labelling. Boundaries are evaluated using the hit-rate metric, which considers a boundary valid if it is approximately equal to an annotation, within a fixed tolerance window. Consistently with MIREX standards [12], tolerances are equal to 0.5s and 3s. The hit-rate is then expressed in terms of Precision, Recall, and F-measure, resulting respectively in $P_{0.5}$, $R_{0.5}$, $F_{0.5}$, P_3, R_3, F_3.

5.1 Practical Considerations

PCA is computed with the scikit-learn [33] toolbox, using the ARPACK solver. NMF is computed using the nn_fac



(a) 0.5 seconds tolerance results ($F_{0.5}$)



(b) 3 seconds tolerance results (F_3)

Figure 6: Segmentation results for the different methods and representation.

Table 1: Results of best-performing methods and state-of-the-art on the RWC-Pop dataset. (*Results: 2015 MIREX contest [12].)

		$P_{0.5}$	$R_{0.5}$	$F_{0.5}$	P_3	R_3	F_3
Compression methods	PCA	60.7%	**66.9%**	63.1%	76.1%	**84.0%**	79.2%
	NMF	57.4%	60.4%	58.3%	73.6%	77.6%	74.9%
	Autoencoder	60.1%	62.6%	60.9%	78.9%	82.5%	**80.1%**
State-of-the-art	Foote [3]	42.0%	30.0%	34.5%	67.1%	47.7%	55.0%
	Spectral clustering [5]	49.2%	45.0%	45.0%	65.5%	60.6%	60.3%
	NTD [9]	58.4%	60.7%	59.0%	72.5%	75.3%	73.2%
	Supervised CNN [13]*	**80.4%**	62.7%	**69.7%**	**91.9%**	71.1%	79.3%
Aligning annotation on downbeats		96.5%	96.2%	96.3%	100%	99.7%	99.9%

toolbox [34]. The AE is developed with Pytorch 1.8.0 [35], trained with the Adam optimizer [36], with a learning rate of 0.001, divided by 10 when the loss function reaches a plateau (20 iterations without improvement) until 1e-5. The optimization stops if no progress is made during 100 consecutive epochs, or after a total of 1000 epochs. The network is initialized with the uniform distribution defined in [37], also known as "kaiming" initialization. Bars were estimated with madmom [17], and spectrograms computed with librosa [19] with default settings, unless specified. All segmentation scores were computed with mir_eval [38].

The entire code for this work is open-source, and contains experimental notebooks for reproducibility [3]. Performing 1000 epochs for a song takes between 1.5 minute (for chromagrams) and 5 minutes (for Mel/Log Mel spectrograms) on an Intel® Core(TM) i7 CPU, NMF takes less than 3 seconds and PCA less than a second.

5.2 Results

Figure 6 presents results on the RWC-Pop dataset for all the methods and the features. The dimension of compression d_c is chosen as the best performing one for the F_3 metric. The Log Mel and Nonnegative Log Mel spectrogram seem to outperform the other features at both tolerances, except for the PCA which obtain similar performances with MFCC. PCA and AE achieve the state-of-the-art level of performance with 3-seconds tolerance, as presented in Table 1. With 0.5-second tolerance, all techniques obtain lower results than those of the state-of-the-art, but outperform the blind state-of-the-art. Results are consistent with those of [23], where PCA obtain similar or better reconstruction errors than linear or shallow autoencoders. Still, we believe that the presented AE can be greatly improved.

NMF obtain lower results than the other methods, but, due to the nonnegativity, it is expected that NMF result in a part-based decomposition of the original barwise spectrogram. The part-based property can lead to interpretable factors, as observed in many applications [21, 39], and is important for pattern uncovering in NTD [9]. Nonetheless, in this study, we do not have quantitative results to support this claim.

Results seem still improvable with 0.5-second tolerance. A wrong estimation at 0.5s can be due to an incorrect frontier estimation, but also to an incorrect bar estimation. Results when aligning the annotation on downbeats presented in Table 1 indicate that bar estimation seems precise on the RWC Pop dataset.

6. CONCLUSION

This article introduce barwise compression schemes, which appear as competitive schemes for Music Structure Analysis. In our experiments on the RWC-Pop music dataset, segmentation scores obtained with compressed representations outperform the results of the blind state-of-the-art, and our best-performing methods reach the level of performance of the global state-of-the-art at 3-seconds tolerance, while being unsupervised. Still, this work focus on the RWC-Pop dataset, and would benefit from further investigations on different musical contexts. In particular, it would be interesting to study the barwise compression schemes in condition where bars are more ambiguous and/or less regular (polyrhythms/polymeters, changing meters, etc) and on more erratic structures (*i.e.* segments less concentrated around the sizes of 4 and 8 bars).

Future work will focus on improving the proposed paradigm, for instance with nonnlinear compression meth-

[3] https://gitlab.inria.fr/amarmore/BarwiseMusicCompression

ods such as kernel methods, or by improving the autoencoder in using strategies such as transfer learning from a song-independent AE or exploring various network architectures or regularizations. Finally, while the convolutional dynamic programming algorithm is competitive, it is unstable to small variations in autosimilarities and should be made more robust.

Altogether, we believe that these first results pave the way to an interesting paradigm using barwise compression algorithms, and notably single-song AEs, for the description of structural elements in music.

7. REFERENCES

[1] J. Paulus, M. Müller, and A. Klapuri, "Audio-based music structure analysis," 2010.

[2] O. Nieto, G. J. Mysore, C.-i. Wang, J. B. Smith, J. Schlüter, T. Grill, and B. McFee, "Audio-based music structure analysis: Current trends, open challenges, and applications," *Trans. Int. Soc. for Music Information Retrieval*, vol. 3, no. 1, 2020.

[3] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *2000 IEEE Int. Conf. Multimedia and Expo. ICME2000. Proc. Latest Advances in the Fast Changing World of Multimedia*, vol. 1. IEEE, 2000, pp. 452–455.

[4] J. Serra, M. Müller, P. Grosche, and J. L. Arcos, "Unsupervised music structure annotation by time series structure features and segment similarity," *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1229–1240, 2014.

[5] B. McFee and D. Ellis, "Analyzing song structure with spectral clustering," in *Int. Soc. Music Information Retrieval (ISMIR)*, 2014, pp. 405–410.

[6] O. Nieto and T. Jehan, "Convex non-negative matrix factorization for automatic music structure identification," in *2013 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 236–240.

[7] I. Theodorakopoulos, G. Economou, and S. Fotopoulos, "Unsupervised music segmentation via multi-scale processing of compressive features' representation," in *2013 18th International Conference on Digital Signal Processing (DSP)*. IEEE, 2013, pp. 1–6.

[8] M. McCallum, "Unsupervised learning of deep features for music segmentation," in *ICASSP*. IEEE, 2019, pp. 346–350.

[9] A. Marmoret, J. Cohen, N. Bertin, and F. Bimbot, "Uncovering audio patterns in music with nonnegative tucker decomposition for structural segmentation," in *ISMIR*, 2020, pp. 788–794.

[10] J.-C. Wang, J. B. Smith, W.-T. Lu, and X. Song, "Supervised metric learning for music structure feature," *ISMIR*, 2021.

[11] J. Salamon, O. Nieto, and N. J. Bryan, "Deep embeddings and section fusion improve music segmentation," *ISMIR*, 2021.

[12] "MIREX wiki," https://www.music-ir.org/mirex/wiki/MIREX_HOME.

[13] T. Grill and J. Schlüter, "Music boundary detection using neural networks on combined features and two-level annotations," in *ISMIR*, 2015, pp. 531–537.

[14] E. Humphrey, J. Bello, and Y. LeCun, "Feature learning and deep architectures: New directions for music informatics," *J. Intelligent Information Systems*, vol. 41, no. 3, pp. 461–481, 2013.

[15] R. Agrawal and S. Dixon, "Learning frame similarity using siamese networks for audio-to-score alignment," in *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 141–145.

[16] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Popular, Classical and Jazz Music Databases," in *ISMIR*, vol. 2, 2002, pp. 287–288.

[17] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proc. 24th ACM Int. Conf. Multimedia*, Amsterdam, The Netherlands, 10 2016, pp. 1174–1178.

[18] I. Vatolkin, F. Ostermann, and M. Müller, "An evolutionary multi-objective feature selection approach for detecting music segment boundaries of specific types," in *Proc. Genetic and Evolutionary Computation Conf.*, 2021, pp. 1061–1069.

[19] B. McFee *et al.*, "librosa/librosa: 0.8.1rc2." Zenodo, May 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4792298

[20] B. McFee and D. Ellis, "Learning to segment songs with ordinal linear discriminant analysis," in *ICASSP*. IEEE, 2014, pp. 5197–5201.

[21] N. Gillis, *Nonnegative Matrix Factorization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2020.

[22] J. Engel *et al.*, "Neural audio synthesis of musical notes with wavenet autoencoders," in *Int. Conf. Machine Learning*. PMLR, 2017, pp. 1068–1077.

[23] F. Roche, T. Hueber, S. Limier, and L. Girin, "Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models," *arXiv preprint arXiv:1806.04096*, 2018.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[26] G. Peeters and G. Richard, "Deep Learning for Audio and Music," in *Multi-faceted Deep Learning: Models and Data*. Springer, 2021. [Online]. Available: https://hal.telecom-paris.fr/hal-03153938

[27] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[28] B. T. Nebgen, R. Vangara, M. A. Hombrados-Herrera, S. Kuksova, and B. S. Alexandrov, "A neural network for determination of latent dimensionality in non-negative matrix factorization," *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 025012, 2021.

[29] J. Paulus, M. Müller, and A. Klapuri, "Audio-based music structure analysis." in *ISMIR*. Utrecht, 2010, pp. 625–636.

[30] G. Sargent, F. Bimbot, and E. Vincent, "A regularity-constrained viterbi algorithm and its application to the structural segmentation of songs," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[31] F. Bimbot, G. Sargent, E. Deruty, C. Guichaoua, and E. Vincent, "Semiotic description of music structure: An introduction to the quaero/metiss structural annotations," in *53rd Int. Conf. Audio Engineering Soc.*, 2014.

[32] O. Nieto and J. Bello, "Systematic exploration of computational music structure research." in *ISMIR*, 2016, pp. 547–553.

[33] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] A. Marmoret and J. Cohen, "nn_fac: Nonnegative factorization techniques toolbox," 2020.

[35] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[36] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Computer Vision*, 2015, pp. 1026–1034.

[38] C. Raffel *et al.*, "mir_eval: A transparent implementation of common MIR metrics," in *ISMIR*, 2014.

[39] D. Lee and H. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

FORMULATING FIRST SPECIES COUNTERPOINT WITH INTEGER PROGRAMMING

Tsubasa Tanaka
Tokyo University of the Arts
tanaka.tsubasa@ms.geidai.ac.jp

ABSTRACT

Counterpoint is a theory for composing western polyphonic music. It consists of many rules that regulate melodic and harmonic relationships within each voice or between voices. Because satisfying all of the rules is difficult, how to search for solutions computationally has been an interesting problem. This paper proposes the first integer-programming-based formulation for the automatic composition of first species counterpoint, the simplest form of counterpoint, as a discrete optimization problem with constraints. Its difference from existing approaches such as constraint programming, stochastic method, and genetic algorithm is that it aims at efficiently accomplishing strict constraint satisfaction and optimality of the solution at the same time.

1. INTRODUCTION

Counterpoint is a theory for composing western polyphonic music. Especially, it has been developed through the vocal polyphony of the Renaissance period and then inherited in the music of later periods. It consists of many rules that regulate melodic and harmonic relationships within each voice or between voices.

Species counterpoint is a step-by-step approach to study counterpoint in the style of Palestrina, a well-known Renaissance composer, adopted in an influential book *Gradus ad Parnassum* of J.J.Fux [1]. There are five species, each of which has a fixed rhythmic structure. Every species has the same given melody called *cantus firmus* (fixed melody in Latin) whose rhythm is monotonous (one whole note in a measure), and students compose the counter-melodies that are compatible with the cantus firmus to complete polyphonic music (Figure 1). The rhythmic structures of the counter-melody of the respective species are as follows:

- First species: one whole note in a measure.
- Second species: two half notes in a measure
- Third species: four fourth notes in a measure.
- Fourth species: one note as *suspension* in an measure (two half notes are tied over consecutive measures)

Copyright: © 2022 Tsubasa Tanaka et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

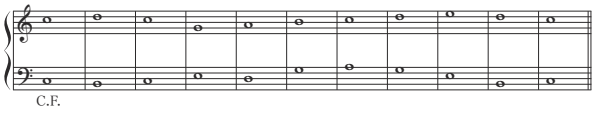


Figure 1. A realization of first species counterpoint from [2]. The lower voice is the cantus firmus.

- Fifth species: all of the previous species are mixed.

When studying counterpoint, composing a solution that does not have any violation of the rules is extremely difficult. Even checking whether a realization satisfies all of the rules is difficult for a student. Because of this, a student needs a human teacher who corrects their realizations. Even for professional teachers, satisfying all of the rules is not easy. Therefore, a computer system that can search for solutions (model answers) or modify the user's realizations would be helpful in the context of music education.

There have been different types of computational research for automatically finding solutions by computer. [3] is one of the earliest researches. It is based on a backtracking algorithm and could only provide solutions with some compromises because it uses a heuristic searching method. [4] is based on constraint programming and could provide exact solutions to small problems. [5] is based on a genetic algorithm in which the constraints are expressed by values that constitute the objective function. This approach has the merit that it can adjust the weights of respective constraints according to their importance. This allows it to find semi-optimal solutions quickly, although the optimal solutions are not assured. [6, 7] are based on a stochastic model to imitate the statistical tendency in existing pieces. They use dynamic programming to search for solutions efficiently, although they have a limitation in that they assume the Markov property that does not hold necessarily.

Following these models, this paper proposes a new computational model for composing first species counterpoint, the simplest form of species counterpoint, based on integer programming. The goal of this model is to accomplish "strict" constraint satisfaction and optimality of satisfying the "soft" and/or "global" constraints at the same time. Here, "soft" constraints are the rules that are less important than "strict" constraints that must be satisfied. Integer programming can formulate both strict and soft constraints. While the strict constraints can be formulated with mathematical equalities and inequalities, soft constraints can be formulated with objective function like in the model

of [5]. "Global" constraints mean constraints such as frequent uses of contrary motions (two melodies move in opposite directions) and conjunct motions (step-wise melodic motion, i.e., semitone and whole tone), which are about the global characteristics. These global constraints cannot be checked easily when assuming the Markov property like in [6, 7]. These can be treated with equalities, inequalities, or objective function in integer programming. The soft and/or global constraints are somewhat linked to the degree of goodness of composition to be optimized. We discuss it in Section 3.5.

Possible drawbacks of the proposed method are relatively large computational time and the complexity of programming many rules. Although it is beyond the scope of this paper to verify to which extent the integer-programming-based method is realistic to formulate more complex counterpoints, it is at least useful for the basic level of counterpoint education. For this purpose, this paper shows an integer-programming-based formulation of first species counterpoint and shows that a solution (a model answer) can be found in a realistic time.

2. INTEGER PROGRAMMING

Integer programming (IP) is a framework to solve discrete optimization problems whose variables are limited to integers. Famous examples of IP problems include the traveling-salesperson problem and the knapsack problem, which are NP-hard problems. Although IP is a powerful tool for finding optimal solutions for complex combinatorial problems, there is a significant applicability limitation. The constraints and objective function must be typically linear. Therefore, it is less flexible than constraint programming. In general, an IP problem can be expressed in the form as follows:

$$\begin{aligned} \text{Maximize } & z = \sum_j c_j x_j \\ \text{subject to } & \sum_j a_{ij} x_j \leq b_i \ \ (i = 1, 2, ..., m) \\ & x_j = 0, 1, 2, ... \ \ (j = 1, 2, ..., n), \end{aligned}$$

where a_{ij}, b_j, c_j are real numbers and x_j are integer variables. The linear inequalities in the middle, which are constraints, define a region surrounded by the hyperplanes. The integer points in this region are feasible solutions. Among these points, the optimal solutions are maximizers of the linear objective function z.

Once the problem is expressed in this form, efficient algorithms, such as the branch-and-bound and branch-and-cut algorithms, can find optimal solutions. Many solvers have been developed to solve IP problems. Recent solvers can deal with huge problems with thousands of variables thanks to the improvement of algorithms and computer performance. Commercial solvers includes Gurobi [8], CPLEX [9], Nuorium Optimizer [10], and free solvers include CBC [11] and GLPK [12]. In the experiment in this paper, CBC, which is the default solver that accompanies the python library PuLP [13], is used.

The aim of this paper is to show that counterpoint (at least first species counterpoint) can be formulated with IP. To do so, some formulation techniques, such as expressing logical operations by combining 0-1 binary variables and *big-M method*, are used as is discussed in Section 3. For further information about integer programming, refer to [14].

In the following section, the rules of first species counterpoint are enumerated, and they are translated into equalities, inequalities, and objective functions that are linear to show that a formulation of counterpoint is possible with IP.

The rules of counterpoint slightly differ from one textbook to another. This paper refers to the textbook of counterpoint [2], which is mainly based on a French textbook [15] but includes a survey on many existing counterpoint textbooks.

3. FORMULATION

3.1 Definition of Constants and Variables

3.1.1 Constants

Cantus firmus, a given melody, is denoted by an array of pitches, *Cf*. For example, we use the following melody in the experiment:

$$\mathit{Cf} = [0, -5, -7, -8, -7, -10, -5, -8, -10, -12].$$

CfUp is an array that indicates whether the melodic direction of *Cf* from bar t to bar $t + 1$ is upward or not. For above *Cf*,

$$\mathit{CfUp} = [0, 0, 0, 1, 0, 1, 0, 0, 0].$$

CfDown is an array that indicates whether the melodic direction of *Cf* at bar t is downward or not. For above *Cf*,

$$\mathit{CfDown} = [1, 1, 1, 0, 1, 0, 1, 1, 1].$$

T is the number of bars in the cantus firmus. For *Cf*,

$$T = length(\mathit{Cf}).$$

The set of possible harmonic intervals, H, is defined as follows:

$$H = \{0, 3, 4, 5, 7, 8, 9, 12\}$$

The set of possible upward melodic intervals, M_+, is defined as follows:

$$M_+ = \{1, 2, 3, 4, 5, 7, 8, 12\}.$$

The set of possible downward melodic intervals, M_-, is defined as follows:

$$M_- = \{-12, -8, -7, -5, -4, -3, -2, -1\}.$$

The set of possible melodic intervals, M, is the union of the two sets:

$$M = M_+ \cup M_-.$$

The set of possible melodic intervals that indicate leaps (melodic motions that are not conjunct), *smallLeaps*, is defined as follows:

$$\mathit{smallLeaps} = M - \{-2, -1, 1, 2\}.$$

The set of possible melodic intervals that indicate large leaps, *largeLeaps*, is defined as follows:

$$\textit{largeLeaps} = M - \{-4, -3, -2, -1, 1, 2, 3, 4\}.$$

The set of bar numbers where *Cf* has a large leap is denoted by *CfLargeLeapT*.

The set of possible pitches in the register of soprano in a diatonic scale, P, is defined as follows:

$$P = \{0, 2, 4, 5, 7, 9, 11, 12, 14, 16, 17, 19, 21\}.$$

The set of time indices of size $T - n$, T_n, is defined as follows:

$$T_n = \{0, 1, 2, \cdots, T - n - 1\} \ \ (n \geq 0).$$

Minimum number of required contrary motions throughout the piece, *MinContrary*, is tentatively set as $T/2$:

$$\textit{MinContrary} = T/2.$$

Minimum number of required conjunct motions throughout the piece, *MinConjunct*, is tentatively set as $T/2$:

$$\textit{MinConjunct} = T/2.$$

3.1.2 Main Variables

0-1 binary variable that indicates whether the harmonic interval of counter-melody at bar t is i or not, $h_{t,i}$, is defined as follows:

$$h_{t,i} \in \{0, 1\} \ \ (t \in T_0).$$

0-1 binary variable that indicates whether the melodic interval of counter-melody from bar t to bar $t + 1$ is i or not, $m_{t,i}$, is defined as follows:

$$m_{t,i} \in \{0, 1\} \ \ (t \in T_1).$$

0-1 binary variable that indicates whether the pitch of counter-melody at bar t is u or not, $p_{t,u}$, is defined as follows:

$$p_{t,u} \in \{0, 1\} \ \ (t \in T_0).$$

Integer variable that indicates harmonic interval between cantus firmus and counter-melody at bar t, $\textit{hInterval}_t$, is defined as follows:

$$\textit{hInterval}_t \in \mathbb{Z} \ \ (t \in T_1).$$

Integer variable that indicates melodic interval of counter-melody from bar t to bar $t + 1$, $\textit{mInterval}_t$, is defined as follows:

$$\textit{mInterval}_t \in \mathbb{Z} \ \ (t \in T_1).$$

3.1.3 Auxiliary Variables

0-1 binary variable that indicates whether the melodic direction of counter-melody is upward or not from bar t to bar $t + 1$, $\textit{up}_t$, is defined as follows:

$$\textit{up}_t \in \{0, 1\} \ \ (t \in T_1).$$

0-1 binary variable that indicates whether the melodic direction of counter-melody is downward or not, $\textit{down}_t$, is defined as follows:

$$\textit{down}_t \in \{0, 1\} \ \ (t \in T_1).$$

0-1 binary variable that indicates whether the melodic motion of counter-melody is conjunct or not from bar t to bar $t + 1$, $\textit{conjunct}_t$ is defined as:

$$\textit{conjunct}_t \in \{0, 1\} \ \ (t \in T_1).$$

0-1 binary variable that indicates whether the melodic motion between cantus furmus and counter-melody is contrary or not from bar t to bar $t + 1$, $\textit{contrary}_t$, is defined as:

$$\textit{contrary}_t \in \{0, 1\} \ \ (t \in T_1).$$

0-1 binary variable that indicates whether the melodic motion of counter-melody from bar t to bar $t + 2$ changes direction at bar $t + 1$ or not, $\textit{turn}_t$, is defined as follows:

$$\textit{turn}_t \in \{0, 1\} \ \ (t \in T_1).$$

Integer variable that indicates the sum of the numbers of leaps of counter-melody at bars t and $t + 1$ (to know whether there is a consecutive leaps or not), $\textit{consecutiveLeap}_t$, is defined as follows:

$$\textit{consecutiveLeap}_t \in \{0, 1, 2\} \ \ (t \in T_2).$$

0-1 binary variable that indicates whether the melodic motion in counter-melody from bar t is a large leap or not, $\textit{largeLeap}_t$, is defined as follows:

$$\textit{largeLeap}_t \in \{0, 1\} \ \ (t \in T_1).$$

Integer variable that indicates the maximum value of pitches in the whole counter-melody, *maxP*, is defined as follows:

$$\textit{maxP} \in \{x \in \mathbf{N} \mid 0 \leq x \leq 21\}.$$

3.2 Basic relationship between variables

There must be constraints between variables defined above. This subsection presents the basic relationship between m, p, and h. Only one of the possible intervals or pitches is realized at bar t. These constraints are expressed by the following equations:

$$\sum_{i \in M} m_{t,i} = 1 \ \ (t \in T_1), \tag{1}$$

$$\sum_{u \in P} p_{t,u} = 1 \ \ (t \in T_0), \tag{2}$$

$$\sum_{i \in H} h_{t,i} = 1 \ \ (t \in T_0). \tag{3}$$

Harmonic and melodic intervals at bar t are expressed by h and m, respectively:

$$\textit{hInterval}_t = \sum_{i \in H} i \cdot h_{t,i} \ \ (t \in T_0), \tag{4}$$

$$\mathit{mInterval}_t = \sum_{i \in M} i \cdot m_{t,i} \ \ (t \in T_1). \tag{5}$$

There must be a consistency between melodic interval and harmonic interval:

$$\begin{aligned}\mathit{mInterval}_t = (\mathit{hInterval}_{t+1} + \mathit{Cf}[t+1]) \\ - (\mathit{hInterval}_t + \mathit{Cf}[t]) \ \ (t \in T_1).\end{aligned} \tag{6}$$

There must be a consistency between two different expressions of pitch in the counter-melody (when $p_{t,u} = 1$, $u \cdot p_{t,u}$ is the pitch at bar t and it must equals to $\mathit{Cf}[t] + \mathit{hInterval}_t$). Therefore,

$$\sum_{u \in P} u \cdot p_{t,u} = \mathit{Cf}[t] + \mathit{hInterval}_t \ \ (t \in T_0). \tag{7}$$

3.3 Preparation of Auxiliary Variables

This subsection presents the constraints for preparing other auxiliary variables.

If one of the melodic interval in M_+ occurs at bar t, the melodic direction at t must be upward, and vice versa for the downward direction. The direction of melody is either up or down (repetition of the same pitches is excluded in advance by M). Therefore,

$$\mathit{up}_t \geq m_{t,i} \ \ (t \in T_1, \ i \in M_+), \tag{8}$$

$$\mathit{down}_t \geq m_{t,i} \ \ (t \in T_1, \ i \in M_-), \tag{9}$$

$$\mathit{up}_t + \mathit{down}_t = 1 \ \ (t \in T_1). \tag{10}$$

When both cantus firmus and counter-melody have the same melodic direction, there must be a contrary motion at bar t ($\mathit{contrary}_t = 1$). Therefore,

$$\mathit{contrary}_t = \mathit{CfUp}[t] \cdot \mathit{down}_t + \mathit{CfDown}[t] \cdot \mathit{up}_t (t \in T_1). \tag{11}$$

When there is a conjunct motion at bar t of counter-melody, $\mathit{conjunct}_t$ must become 1. Therefore,

$$\begin{aligned}\mathit{conjunct}_t = m_{t,1} + m_{t,-1} + m_{t,2} + m_{t,-2} \\ (t \in T_1).\end{aligned} \tag{12}$$

$\mathit{consecutiveLeap}_t$ is the number of consecutive (small) leaps at bar t and $t+1$ in counter-melody. Therefore,

$$\mathit{consecutiveLeap}_t = \sum_{i \in \mathit{smallLeaps}} (m_{t,i} + m_{t+1,i}) \ \ (t \in T_2). \tag{13}$$

$\mathit{largeLeap}_t$ indicates whether there is a large leap or not at bar t in counter-melody. Therefore,

$$\begin{aligned}\mathit{largeLeap}_t = \sum_{i \in \mathit{LargeLeaps}} m_{t,i} \\ (t \in \mathit{CfLargeLeapT}).\end{aligned} \tag{14}$$

There must be consistency between variables *up*, *down*, and *turn* (for example, if $\mathit{up}_t = 1$ and $\mathit{down}_{t+1} = 1$, turn_t must be 1. And so on.). Therefore,

$$\mathit{up}_t + \mathit{down}_{t+1} \leq 1 + \mathit{turn}_t \ \ (t \in T_1), \tag{15}$$

$$\mathit{down}_t + \mathit{up}_{t+1} \leq 1 + \mathit{turn}_t \ \ (t \in T_1), \tag{16}$$

$$\mathit{up}_t + \mathit{up}_{t+1} \leq 1 + (1 - \mathit{turn}_t) \ \ (t \in T_1), \tag{17}$$

$$\mathit{down}_t + \mathit{down}_{t+1} \leq 1 + (1 - \mathit{turn}_t) \ \ (t \in T_1). \tag{18}$$

3.4 Rules of Counterpoint

3.4.1 Limited use of harmonic interval 0.

Harmonic interval 0 is possible only at the first bar ($t = 0$) and the last bar ($t = T - 1$). Therefore,

$$h_{t,0} = 0 \ \ (t \in 1, 2, 3, ...T - 2). \tag{19}$$

3.4.2 Prohibitions of parallel fifths and octaves

Consecutive occurrence of harmonic intervals of fifth and that of eighth are prohibited. Therefore,

$$h_{t,i_1} + h_{t+1,i_2} \leq 1 \ \ (t \in T_1, i_1, i_2 \in \{0, 12, 24\}), \tag{20}$$

$$h_{t,i_1} + h_{t+1,i_2} \leq 1 \ \ (t \in T_1, i_1, i_2 \in \{7, 19\}). \tag{21}$$

3.4.3 Prohibitions of hidden fifths and octaves

Harmonic motions that reach fifth or eighth in a parallel motion are prohibited. (For example, when both voices go up, $\mathit{up}_t = 1$ and this must imply $h_{t+1,24} = 0$, which means harmonic interval 24 does not occur.) Therefore,

if $\mathit{CfUp}[t] == 1$ $(t \in T_1)$,

$$h_{t+1,24} \leq 1 - \mathit{up}_t, \tag{22}$$

$$h_{t+1,19} \leq 1 - \mathit{up}_t, \tag{23}$$

$$h_{t+1,12} \leq 1 - \mathit{up}_t, \tag{24}$$

$$h_{t+1,7} \leq 1 - \mathit{up}_t, \tag{25}$$

$$h_{t+1,0} \leq 1 - \mathit{up}_t. \tag{26}$$

else if $\mathit{CfDown}[t] == 1$,

$$h_{t+1,24} \leq 1 - \mathit{down}_t, \tag{27}$$

$$h_{t+1,19} \leq 1 - \mathit{down}_t, \tag{28}$$

$$h_{t+1,12} \leq 1 - \mathit{down}_t, \tag{29}$$

$$h_{t+1,7} \leq 1 - \mathit{down}_t, \tag{30}$$

$$h_{t+1,0} \leq 1 - \mathit{down}_t. \tag{31}$$

3.4.4 Prohibited arpeggio patterns of triads

In consecutive three notes in counter-melody, arpeggio of triads (major, minor, diminished, and augmented) in one direction (without turn) must not appear. Therefore,

$$m_{t,3} + m_{t,4} + m_{t+1,3} + m_{t+1,4} \leq 1 \ \ (t \in T_2), \tag{32}$$

$$m_{t,-3} + m_{t,-4} + m_{t+1,-3} + m_{t+1,-4} \leq 1 \ \ (t \in T_2), \tag{33}$$

$$m_{t,3} + m_{t,4} + m_{t+1,5} \leq 1 \ \ (t \in T_2), \tag{34}$$

$$m_{t,-5} + m_{t+1,-3} + m_{t+1,-4} \leq 1 \ \ (t \in T_2), \tag{35}$$

$$m_{t,5} + m_{t+1,3} + m_{t+1,4} \leq 1 \ \ (t \in T_2), \tag{36}$$

$$m_{t,-3} + m_{t,-4} + m_{t+1,-5} \leq 1 \ \ (t \in T_2). \tag{37}$$

3.4.5 Prohibition of two consecutive leaps in the same direction

If there are consecutive leaps ($\mathit{consecutiveLeap}_t = 2$), they must form a turn ($\mathit{turn}_t = 1$). And if $\mathit{turn}_t = 0$, $\mathit{consecutiveLeap}_t$ must be less than 2 (i.e., no consecutive leaps at t). Therefore,

$$\mathit{consecutiveLeap}_t \leq 1 + \mathit{turn}_t \ \ (t \in T_2). \tag{38}$$

3.4.6 Prohibition of three-times repetitions of the same pitch in five consecutive notes

There must not be three same pitches in five consecutive notes (repetition of the same pitch in consecutive two notes has been already excluded because M does not include 0.). Therefore,

$$\sum_{s=t}^{t+4} p_{s,u} \leq 2 \ \ (t \in T_4, u \in P). \tag{39}$$

3.4.7 Prohibition of simultaneous large leaps between two voices

When $t \in$ *CfLargeLeapT*, if there is a large leap at t (*largeLeap*$_t = 1$), there must not be a parallel motion at t (*contrary*$_t$ must be 1) to avoid simultaneous parallel leaps between two voices. And if there is a parallel motion at t (*contrary*$_t = 0$), there must not be a large leap at t in counter-melody (*largeLeap*$_t = 0$) to avoid simultaneous large leaps. Therefore,

$$\textit{largeLeap}_t \leq \textit{contrary}_t \ (t \in \textit{CfLargeLeapT}). \tag{40}$$

3.4.8 Large leaps must be preceded or succeeded by opposite melodic movement

This rule is unnecessary because the rule 3.4.5 is sufficient.

3.4.9 In two-voice counterpoint, it is desirable to avoided voice crossing

In this paper, this rule is unnecessary because voice crossing is avoided by the definition of H.

3.4.10 Consecutive parallel movement of third or sixth is limited to three times

The number of consecutive parallel movements of the same harmonic intervals concerning third and sixth must be at most 3. Therefore,

$$\sum_{s=t}^{t+3} (h_{s,3} + h_{s,4}) \leq 3 \ \ (t \in T_3), \tag{41}$$

$$\sum_{s=t}^{t+3} (h_{s,8} + h_{s,9}) \leq 3 \ \ (t \in T_3), \tag{42}$$

$$\sum_{s=t}^{t+3} (h_{s,15} + h_{s,16}) \leq 3 \ \ (t \in T_3), \tag{43}$$

$$\sum_{s=t}^{t+3} (h_{s,20} + h_{s,21}) \leq 3 \ \ (t \in T_3). \tag{44}$$

3.4.11 Rules for desired melody

According to the textbook we refer to, the criteria for desired melody are as follows:

1. Contrary motions should be used as much as possible.
2. Long sequences of conjunct motion are included.
3. The melody has a "will" as a whole.
4. The voice had better stay within the adequate register of the part.

One way to interpret the first and second points is to regard them as maximizations of the number of contrary motions and that of conjunct motions, which is easy to implement as an objective function described in the next section. Another possible interpretation, which is compatible with the first one, is to introduce minimum thresholds for the number of contrary motions (*MinContrary*) and that of conjunct motions (*MinConjunct*). They can be implemented as the following constraints:

$$\sum_{t \in T_1} \textit{contrary}_t \geq \textit{MinContrary}, \tag{45}$$

$$\sum_{t \in T_1} \textit{conjunct}_t \geq \textit{MinConjunct}. \tag{46}$$

The fourth point can be satisfied by setting P, possible pitches for counter-melody. The third point is the most difficult to understand. One way to interpret it is to introduce a climax (the highest pitch) at unique t, and to suppress the number of zigzag movements (turns) that look like "hesitating." The latter can be implemented as a minimization of the number of turns, which is described in the next section. The former can be implemented by the following constraints:

$$\sum_{t \in T_0} \textit{climax}_t = 1, \tag{47}$$

$$\begin{aligned} &(\textit{Cf}[t] + \textit{hInterval}_t) + (1 - \textit{climax}_t) \\ &\leq \textit{maxP} \ \ (t \in T_0), \end{aligned} \tag{48}$$

$$\begin{aligned} \textit{maxP} \leq\ &(\textit{Cf}[t] + \textit{hInterval}_t) \\ &+ \textit{Width} \cdot (1 - \textit{climax}_t) \ \ (t \in T_0). \end{aligned} \tag{49}$$

Here the last inequality uses the technique of Big-M method that can switch the inequality. If *climax*$_t = 1$, the term of *Width* (this constant is sufficiently "big") is eliminated and the pitch at bar t, *Cf*$[t]$ + *hInterval*$_t$, coincides with *maxP*. Therefore, the climax is realized at bar t. If *climax*$_t = 0$, the inequality does not work as a constraint because *Width* is sufficiently large.

3.5 Objective Function

As we discussed the rule 3.4.11, we can consider that there are three elements that influence the goodness, namely, the numbers of contrary motions, conjunct motions, and turns. Therefore, a way to define the criteria as an objective function is to add or subtract these numbers as follows:

Minimize

$$\sum_{t \in T_2} \textit{turn}_t - \sum_{t \in T_2} \textit{contrary}_t - \sum_{t \in T_2} \textit{conjunct}_t.$$

Of course, we can consider introducing coefficients for these three criteria. Also, there may be other possible criteria. The best choice of the coefficients and the criteria may differ from one user to another. When this proposed

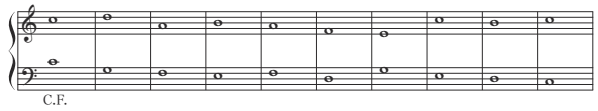



Figure 2. A generated result. The lower voice is cantus firmus, a given melody used in the textbook [2].

criterion	occurrence
turn	6
contrary motions	6
conjunct motions	6

Table 1. The values of the three criteria of the objective function for the piece in Figure 2

method is used in an actual application, some choices may be provided to the users so that they can find their preferences for the objective function. In any case, we assume that the possible criteria can be expressed as linear functions by setting adequate coefficients on them. Therefore, we adopt the above objective function tentatively for simplicity of discussion without loss of generality.

Therefore, integer programming can be applied to this problem because of this assumption and the linearity of the constraints. The constraints define possible solutions that satisfy the strict rules of counterpoint, and the object function defines the optimality within them from the chosen criteria for the goodness of composing counter-melody.

3.6 Generated Result

The formulation of the previous section was implemented with PuLP, a python library that can model linear optimization problems. CBC, the free default solver of PuLP was used to search for the solution. The type of computer was MacBook Pro 2016 with 2.6 GHz quad-core Intel Core i7.

Figure 2 shows an optimal solution found. It took 3.88 seconds to search for it. The value of the objective function is -6. The contribution of each criterion is shown in Table 1. From this table, we can see that three criteria are equally satisfied in a good balance. The climax of the counter-melody is realized at bar 2. Thanks to the rule of climax in 3.4.11, the melody has a certain degree of directionality as a melody (hopefully a "will" described in 3.4.11). The melody goes down and then goes up in the last part. Thanks to the modest number of turns and relatively many contrary and conjunct motions, the counter-melody looks stable and independent from the cantus firmus. For comparison, another generated result is shown in Figure 3. In this case, only the number of turns was maximized to observe the importance of the three criteria. However, because of the large number of turns and less conjunct motions, the counter-melody looks relentless, and its unity as a melody looks a little broken. The original objective function seems to work well from this comparison.

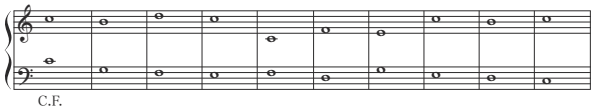


Figure 3. A piece generated with a different objective function. Only the number of turns was taken into account as the objective function (maximized). In this piece, the number of turns, contrary motions, and conjunct motions are 8, 6, 5, respectively.

4. CONCLUSIONS

This paper presented a new formulation of first species counterpoint based on integer programming. The rules of counterpoint were expressed by linear constraints and objective function so that integer programming could be applied. As a result, optimal solutions were found within a realistic time. There are remaining questions. Although counterpoint consists of explicit rules in the textbook, are they enough to create excellent pieces? Are the generated pieces natural enough to the extent that listeners can not distinguish whether they are composed by humans or by a computer? To answer these questions, an evaluation of generated pieces and a comparison between the generated pieces and human composers' pieces would be necessary. Also, formulating counterpoints of more complex species and larger numbers of voices is a future task.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 19K13024.

5. REFERENCES

[1] J. J. Fux, *The study of counterpoint from Johann Joseph Fux's Gradus ad Parnassum*. WW Norton & Company, 1965.

[2] Y. Hiroshi, *Traité de contrepoint d'après les méthodes du Conservatoire de Paris*. Ongakunotomosha, 2012.

[3] W. Shottstaedt, "Automatic counterpoint," in *Current directions in computer music research*, 1989.

[4] A. Torsten, C. Anagnostopoulou, and M. Alcorn, "Strasheela: design and usage of a music composition environment based on the Oz programming model," in *Proceedings of International Conference on Multiparadigm Programming in Mozart/Oz*, Charleroi, 2005, pp. 277–291.

[5] D. Herremans and K. Sörensen, "Composing first species counterpoint with a variable neighbourhood search algorithm," in *Journal of Mathematics and the Arts 6.4*, 2012, pp. 169–189.

[6] M. Farbood and B. Schöner, "Analysis and synthesis of Palestrina-style counterpoint using Markov chains," in *Proceedings of International Computer Music Conference*, 2001.

[7] T. Tanaka and et al., "Automatic music composition based on counterpoint and imitation using stochastic models," in *Proceedings of 7th Sound and Music Computing Conference*, Barcelona, 2010, pp. 213–218.

[8] https://www.gurobi.com/.

[9] https://www.ibm.com/products/ilog-cplex-optimization studio.

[10] https://www.msi.co.jp/nuopt/index.html.

[11] https://github.com/coin or/Cbc.

[12] https://www.gnu.org/software/glpk/.

[13] https://coin or.github.io/pulp/index.html.

[14] D. S. Chen, R. Batson, and Y. Dang, *Applied Integer Programming: Modeling and Solution*. Wiley, 2010.

[15] G. Noël and B. Marcel, *Traité de contrepoint*. Durand, 1964.

GUITAR IMPROVISATIONS WITH HEXAPHONIC MULTIEFFECT (GIHME) DATASET AND PRACTICE ANALYSIS

Loïc Reboursière
ISIA Lab
UMONS
loic.reboursiere@umons.ac.be

Thierry Dutoit
ISIA Lab
UMONS
thierry.dutoit@umons.ac.be

Vincent Tiffon
PRISM
Aix Marseille Université
tiffon@prism.cnrs.fr

ABSTRACT

This paper presents a new guitar dataset made out of richly annotated guitarist improvisations. The annotations include notes, playing techniques, instrument tuning, audio effects configurations as well as transcriptions of post improvisation interviews. The dataset gathers ten hours of improvisations and around five hours of interviews. These accompanying data make this dataset suitable for a variety of research domains : from MIR to musical improvisation analysis and musicology. The recordings that have yielded to this dataset were done in the context of an hexaphonic multieffect practice study. This hexaphonic multieffect is meant to work with an hexaphonic guitar (one pickup per string guitar) and grants the player with independent audio effects configurations for each string. This paper presents the dataset, and the experiment it has been gathered from. It also details, based on the transcriptions of the interviews, a first analysis of the specificities of an hexaphonic multieffect practice.

1. INTRODUCTION

An hexaphonic guitar is an electric (or acoustic) guitar equipped with an hexaphonic pickup. This device gathers six individual pickups, one per string. As a result, six audio signals are available for further processing [1] . With such a system, different audio effects or analysis tools can be applied to each string independently. This type of pickup appeared in the late 1970s with the guitar synthesizer which as its name implies corresponds to a guitar that can control a synthesizer. On this type of instrument, hexaphonic pickups are of great help in converting notes to control signals, because they narrow down the complexity of pitch detection from polyphonic to six monophonic algorithms running in parallel. Apart from that main commercially developed use case, hexaphonic pickups can be used for independent-string audio processing. The first physical units integrating independent-string audio processing (either in synthesizers or in guitar pedal effects) appeared at the end of the 1970s but, despite the large amount of creative potential it seems to have, hexaphonic audio processing has never reached a larger audience and remained a niche. Interests in such an approach have resurfaced in recent years with companies like Cycfi [2] or Synquanon [3] . The former manufactures the Nu Series pickups [4] . The Nu is a one-string pickup active pickup which can be gathered in a series of any number (i.e. it can be adapted to the bass guitar, double bass, 6, 7 or 8-string guitar, etc.). The latter develops hexaphonic audio effects in the form of Eurorack modules.

The series of experiments and the resulting dataset [5] presented here were made to try to understand how individual string processing can modify (or not) the practice of guitarists. Five guitarists used the hexaphonic multieffect for three days to record improvisations based on predefined scenarios. The resulting dataset is made out of objective data (notes, playing techniques, effect configurations, etc.) which were collected automatically, semi-automatically or by hand, and subjective data (transcriptions of guitarist interviews). These different types of data make the dataset suitable for different research fields ranging from Music Information Retrieval (MIR) to musicology as well as music improvisation analysis. Ten hours of musical improvisations are being annotated and four and a half hour of interviews have been transcribed. As the amount of objective data is quite important (despite the limited number of guitarists), the first analysis presented here is based on interviews of guitarists. This analysis helps to highlight terms and notions describing different points of interest of such a setup for the musicians.

While section 2 covers works related to our topic, sections 3, 4 and 5 develop different points of the experiment. Section 6 describes further the built dataset and section 7 brings an analysis based on the interviews of the guitarists. Finally, section 8 highlights future work before the conclusion (section 9).

2. RELATED WORKS

Music datasets are mostly built and used in MIR-related contexts. Some contains multiple types of musical instru-

[1] As a comparison, on regular electric guitars, monophonic pickups mix the sound of all resonating strings down to one audio signal.

Copyright: © 2022 Loïc Reboursière et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[2] https://www.cycfi.com/ (08/02/2022).
[3] https://www.synquanon.com/ (08/02/2022).
[4] https://www.cycfi.com/projects/nu-series/ (08/02/2022).
[5] https://github.com/numediart/GIHME (08/02/2022).

ments [1], and some are oriented towards specific instruments such as the piano [2, 3] or the guitar [4, 5]. Most of these datasets are built for pitch and onset detection but some of them are made for tasks such as guitar playing techniques [6], chords [7], effects [8] and playing modes [9] detection. Other approaches such as non-negative matrix factorization has been used with the hexaphonic guitar [10] to provide automatic transcription of the instrument.

Aside from these datasets which were built to improve existing results on the MIR-related tasks, some are built to study musical and instrumental practices such as the Weimar Jazz Database from the Jazzomat project [11]. This project uses, as well as other tools, different MIR techniques to extract relevant information to describe and classify jazz improvisations. As mentioned above, the dataset presented in this paper goes a bit further by making available to the community recordings of guitarists' improvisations made using an hexaphonic multieffect and different types of annotations.

First hexaphonic effects appear in the late 1970s and the beginning of the 1980s. The ARP Avatar [6] guitar synthesizer and the Roland GR-100 [7] both include an hexaphonic distortion effect while the GR-300 [8] integrates an hexaphonic harmonizer. Apart from these already-established companies, engineer Keith McMillen developed the PolyFuzz [9] guitar pedal (used by guitarist John Abercrombie) which integrates distortion, fuzz and suboctave effects. The swiss luthier Matthias Grob built, under the brand Paradis and around the same period, its PolyDistortion [10] followed by a multieffect. He then went on developing the Polysubbass [11] (which octave down each string) and the Mathons VST plugins series [12]. More recently, several scientific works dealt with the development of hexaphonic audio processing tools [12] and their use in performance context [13–15] [13].

3. EXPERIMENT STRUCTURE

The experiment presented here was built up to investigate specific uses of hexaphonic pickup and multieffect, namely the distribution of audio effects on specific groups of strings and the control of individual bypass of the audio effects. For this experiment, five guitarists have been recorded playing the hexaphonic multieffect on specific predefined scenarios. Four of them are professional guitarists and/or composers and/or improvisers and the last one is an amateur guitarist/composer [14]. The four professional guitarists are part of a collective of musicians [15] whose musical projects range from jazz and improvisation, to contemporary and prepared instruments while the amateur guitarist mostly evolves in rock style-related music. While the practice of the professional guitarists can vary greatly from one guitarist to another, one could say that they mainly share the same modes of playing as a common ground. As one may have noticed, this dataset is clearly unbalanced in terms of professionals vs. amateurs. The point of this first analysis was not to compare guitarists in their use of the hexaphonic setup, but to try to understand how one guitarist practice is impacted (or not) by the use of this setup. In the rest of the paper, we'll be referring to the guitarists using number from 1 to 5, 5 being the amateur guitarist.

3.1 Pre-Defined Scenarios

The whole process of this experiment go through the three following scenarios:

- The first scenario is a "discovery scenario" where the guitarist, with the aid of the researcher, tests each hexaphonic effect, builds presets and create a "global sound" (i.e. chain of chosen effects and presets) to start working with on the next scenario. No improvisation was recorded during this scenario.
- The second scenario investigates the distribution of effects on specific guitar strings by applying the chosen "global sound" to different groups of strings while the remaining strings are left "dry" (i.e. without any effects).
- The third scenario studies how this setup can be controlled in a performance context. A generic MIDI foot controller (Behringer FCB1010 [16]) was used to control different granularities of individual audio effects bypass controls (e.g. global effect bypass vs. string-independent bypass).

As a matter of clarity, the "discovery scenario" was named scenario 0 and the two remaining, respectively, scenario 1 and scenario 2.

3.2 Scenarios 1 and 2 Protocol

Scenarios 1 and 2 follow the same protocol:

- They are made out of five sub-scenarios;
- Each sub-scenario is made of three steps:
  - Test : the guitarist plays with the proposed subscenario's configuration and tries to develop musical ideas that can be used during the improvisation;

[6] http://www.vintagesynth.com/arp/avatar.php (08/02/2022).
[7] https://www.joness.com/gr300/GR-100.htm (08/02/2022).
[8] https://www.joness.com/gr300/GR-300.htm (08/02/2022).
[9] https://dokumen.tips/amp/documents/keith-mcmillen-timeline.html (08/02/2022).
[10] http://www.matthiasgrob.org/pEE/sndhist.htm (08/02/2022).
[11] http://www.polybass.com (08/02/2022).
[12] https://www.mathons.com/ (08/02/2022).
[13] Sound recordings of hexaphonic effects can be found online, https://soundcloud.com/medicationtime/sets/hexaphonic-effects (08/02/2022).
[14] https://www.youtube.com/channel/UCLHyrUsYR-gE5r_4Vs45xkQ (08/02/2022).
[15] https://muzzix.info/ (08/02/2022).
[16] This controller provides ten buttons (configured with on/off behaviours) which can be linked to different mapping configurations (also called pages or banks in this type of devices). The browsing (going up or down) of this list of configurations can be done by two other buttons. Two continuous foot controller are also present on the device but were not used in the context of these experiments.

Name	Strings with effects	Strings with no effect
1_1	E-A-D	G-B-e
1_2	G-B-e	E-A-D
1_3	E-D-B	A-G-e
1_4	A-G-e	E-D-B
1_5	Distribution chosen by the guitarist	

Table 1. Scenario 1 sub-scenarios [20] .

  - Record : once the guitarist is satisfied with its findings, he records an improvisation of 3 to 4 minutes minimum (the longest being 18 minutes);
  - Interview : the guitarist and the researcher talk about specific elements of the improvisation that was just recorded.

- The last two steps can be repeated any number of times the guitarist feels like trying out different presets (i.e. changing the "global sound" to fit one specific sub-scenario) and/or different modes of playing;

- While the first four sub-scenarios are predefined, the fifth one is created by the guitarist. In scenario 1, the guitarist can choose the group of strings on which the "global sound" can be applied. In scenario 2, the guitarist can decide which bypass control configuration he wants and create presets.

Table 1 and 2 summarize the different configurations of the sub-scenarios contained in each scenario. While sub-scenarios 1_1 and 1_2 uses separation between "low" and "high" strings which is already used by guitarists in different playing styles [17] , scenarios 1_3 and 1_4 are relatively unnatural for guitarists as effects are being applied on non-adjacent strings. It also has to be noted that sub-scenarios 2_3 and 2_4 come out of the foot controller structure. Indeed, as only ten buttons [18] are available at once on the controller, mapping strategies needed to be defined in order to access the 36 individual bypasses of the multieffect (6 effects available on 6 strings). Another setup, e.g. with 2 Voes MX-18 (matrix of 6x3 foot switches) controllers [19] (which we didn't know of at the time of the experiment), may have only needed one sub-scenario to access the 36 individual bypass controls.

4. HEXAPHONIC MULTIEFFECT

The hexaphonic multieffect used in these experiments is depicted in figure 1. It has been developed using Cycling' 74 Max software. This patch is an adapted version of the one presented in [13]. The multieffect gathers four main

Name	Bypass controls mapping
2_1	1 button controls the bypass of 1 hexaphonic effect on all strings
2_2	1 button controls the bypass of the effects applied on 1 string
2_3	1 bank per effect and 1 button per string
2_4	1 bank per string and 1 button per effect
2_5	Distribution (2_3 or 2_4) chosen by the guitarist and definition of recallable presets

Table 2. Scenario 2 sub-scenarios.

elements: six hexaphonic audio effects, a bypass matrix grouping individual effects bypass, a routing system to define effects order, an output mixer to adjust strings output levels individually. In order to help guitarists focus on the hexaphony practice as much as possible, the six audio effects (overdrive, delay, ring modulation, flanger, tremolo, reverb) were chosen, arguably, among the most common for electric guitarists. Each of the effects integrates six instances (depicted by graphical colourful sliders) of each of its parameters, a preset system (integrating an interpolation option not used in these experiments) and individual bypasses. Those individual bypasses are gathered and developed graphically (bottom right part of the patch), forming a 6x6 graphical matrix that gives a more convenient visual feedback to the guitarist when using the foot controller in scenario 2. The default order of the effects routing system follows the graphical display of the patch (same as listed above). Only one of the guitarists changed this default setting by moving the delay just before the reverb effect. Lastly, an output mixer is used to balance the sound of each string. This mixer is particularly useful in the first scenario, where the difference in sound amplitude between the strings with effects and the strings without effects can be pretty significant.

5. AUDIO SETUP AND RECORDING PROCESS

The audio processing setup is made out of several components:

- Two Godin guitars equipped with RMC hexaphonic piezoelectric pickups were at the disposal of the guitarists. The first one, a Godin Multiac, is a nylon-string guitar whereas the second one, a Godin LGXT, is a steel-string guitar. Only one of the guitarists tried the first one, but eventually felt more at ease with the second one;

- A homemade breakout box is used to power pickup electronics and to convert the standard 13-pin din connector used for the hexaphonic pickup [21] to six standard monophonic 6.35 mm female jack cable;

- The breakout box is connected to an RME Fireface UCX soundcard through six mono jack cables. The

[17] We can think, e.g., of acoustic blues style where guitarists often plays the accompaniement part on the low strings (often with alternating bass technique) and the melody on the high strings.

[18] See footnote 16 .

[19] `https://www.voes.be/mx18.html` (08/02/2022).

[20] The standard tuning of a six-string guitar is E-A-D-G-B-e, the **E** string being the 6th string and the **e** string, the 1st string.

[21] This "standard" connector is mainly used by Roland which has been the main company developing guitar synthesizers.

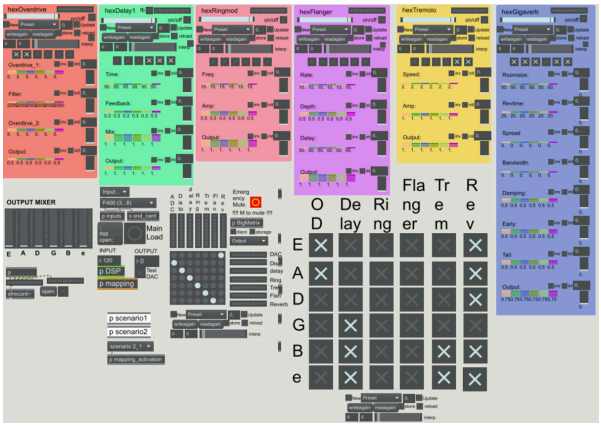


Figure 1. Hexaphonic multieffect used for the experiments.

sampling rate was set to 44100 Hz and 16 bits of precision were used;

- The computer is running Max software and the hexaphonic multieffect patch described in section 4;
- Hexaphonic outputs are then connected to a mixing desk and to a quadriphonic Genelec speaker system (complemented by a subwoofer). This diffusion system was the one present in the recording room and couldn't be moved. No specific spatialization was used as it was not the subject of this experiment. The resulting sounds of the six strings were all equally broadcast on each speaker.

Multiple audio and video files were recorded during the experiment. On the audio side, the hexaphonic audio signals were recorded before ("dry" or "clean") and after ("wet" [22]) the chosen audio effects. As the result, the clean hexaphonic signals permit to easily use any type of algorithm detection to ease the annotation part and the wet hexaphonic signals give a resulting sound with great details. A monophonic reduction of the hexaphonic wet signals was also recorded in order to have a low quality format easier to work with (only the first two channels of hexaphonic audio files can be heard when listened to with classic audio player software). Moreover, the guitarist was asked to start (when ready) a mono mix recording of its improvisation. On the video side, a Nikon 5D mark IV camera was used to capture the guitarists while they were playing (during test, improvisation or interview parts). Video files were recorded with a resolution of 1920x1080 at 25 fps. The screen of the computer, on which the Max patch ran, was also recorded in order to keep track of the user changes to the patch's GUI. The video conference software Zoom was used for that purpose so that recordings could be launched remotely without having the guitarist to do it.

All recorded signals were synced by sound. The guitarist was asked to pluck the lowest string with a palm-muting technique in order to record on each media a sharp event that can easily be detected and referred to.

[22] The "wet" term is often present in effects like reverb for example where levels of modified signals and non modified signals can be modified. In this context, this term refers to the level of the modified signal.

6. DATASET

Once all those raw signals are recorded, several steps are necessary in order to obtain a useable dataset.

The first step was to extract from recorded audio and video files, the media corresponding to each specific subscenario recordings. Syncing points (made by plucking the palm-muted lowest string), as well as timings of subscenario parts (test, recording or interview) were annotated by hand in order to automatically trim and export files corresponding to each sub-scenario. The trim and export steps were done using `ffmpeg` [23] and `sox` [24] software, respectively, for dealing with video and audio files. Each part of each sub-scenario is then made out of five files:

- Two video files: one from the camera, one from the computer screen;
- Three audio files: clean and wet hexaphonic signals as well as monophonic mix down of wet hexaphonic signals.

The second step was to annotate each sub-scenario. Different types of annotations were used for different types of collected data:

- Global information, such as scenario and sub-scenario number, instrument tuning (and its evolution if necessary), duration of the specific part, used audio effects and presets, were collected;
- Played notes have been retrieved by using a semi-automatic method similar to the one used in [5]. Six automatic pitch extractions were performed in parallel, on the hexaphonic clean signals. As tuning (i.e. pitches of the strings when no notes are fretted) of the guitar was known, fret number was inferred from pitch extraction result and string number. `Aubio` library's [25] implementation of `yin-fft` algorithm [16] has been used to perform this task [26] . The data generated by the algorithm have then been manually verified in order to consolidate notes and frets information;
- The activation periods of each individual effects from scenario 2 have been automatically extracted from computer screen videos. A computer vision algorithm has been built to track the changes in the individual bypasses of hexaphonic effects of the graphical user interface (those changes have then been computed as time segments). Moreover, in order to be able to easily visualize those changes with the improvisation, the global bypass matrix from the computer screen videos has been cropped and overlaid on top of the camera video;
- Common guitar playing techniques, such as bend, harmonics, hammer-on, pull-off (see, e.g. [6], for

[23] `https://www.ffmpeg.org/` (08/02/2022).
[24] `http://sox.sourceforge.net/` (08/02/2022).
[25] `https://aubio.org/` (08/02/2022).
[26] It has to be noticed that different pitch detection algorithms were tested inside of Sonic Visualizer software and that the `yin-fft` algorithm appeared to give the best results.

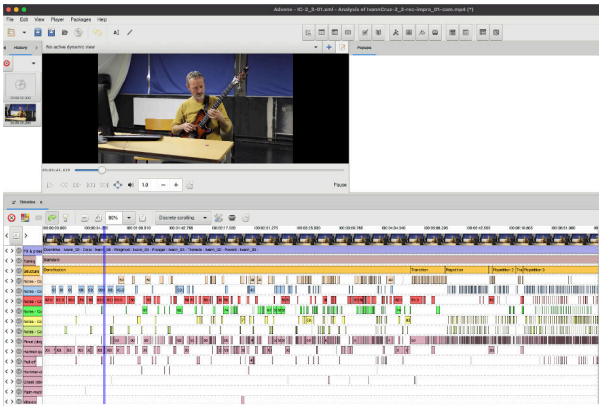


Figure 2. GIHME annotations integration example in Advene software.

details on such techniques), have been manually annotated. Extended playing techniques like the use of a bow,objects or preparations as well as more advanced techniques like *scordatura* [27] are also including in the annotations. Definitions and explanations of those kinds of extended or advanced techniques can be found in [17], [18] and [19] or [20]. To the best of our knowledge, these kinds of techniques are most often not present in literature's datasets.

All the data detailed above are stored in various file formats (`json`, `txt` or `csv`) in order to be compatible with different visualization software like Sonic Visualizer [28] or Advene [29]. An example of the collected data integrated in Advene software can be seen on Figure 2. It has to be noticed that at the time of the submission, the process of acquiring all the objective annotations is not finished.

To complete the objective collected data, interviews made after each improvisation have been transcribed. The method used for the transcription is the one of the verbatim. The use of the verbatim here is made so that researchers from other fields willing to analyse those texts can access the data in minimally modified form. Only some common French oral language words or expressions have been rewritten with more readable terms. The analysis that we are making in the next section is specifically based on those transcriptions.

7. INTERVIEWS ANALYSIS

This analysis of the words of the guitarists is a first step towards understanding and characterizing how their practices have been altered by the use of the a hexaphonic multieffect. The different points presented below give good hints about the impact of hexaphonic multieffect on practices of guitarists, but, for now, it cannot be easily generalizable as the number of guitarists who participate in that experiment is pretty small.

7.1 Constraints and Limitations

Before talking about the impact of the hexaphonic multieffect itself, one common element to all guitarists is that, at some point, they felt constrained. The limitations they endured were most of the time either due to the configurations of the scenarios or to the technical elements used for the experiment.

It has to be noticed that the scenarios and sub-scenario configurations do act as constraints already, as they put all the guitarists in unusual situations, but most of the time the guitarists have managed to work with them and felt stimulated by them. The constraints listed below are the ones where guitarists had a hard time integrating those configuration in their improvisations.

Regarding to the hexaphonic multieffect, several constraints were mentioned : the balance between the volume of dry strings and strings on which effects were applied in scenario 1 was, for example, mentioned by guitarists 1 and 3 as being problematic (the issue was resolved by adding the output mixer mentioned in Section 4). Some effects configurations were felt uneasy : guitarist 1 had trouble playing with different delay times when those were not rhythmically related and guitarist 5 felt that applying distortion only on specific strings was not a natural fit for him.

On the scenario level, all the manipulations needed to access the different individual bypasses in scenarios 2_3, 2_4 and 2_5 (due to the use of a foot controller FCB1010 as mentioned above) was mentioned by guitarists 1, 4 and 5 as being not intuitive and adding a strong inertia to the whole process. Guitarist 2 felt a strong constraint with effect distribution in scenario1_3 and 1_4:

> I was a bit helpless in fact with a setup like this one. [...] I have my guitarist's reflexes that try to find something but which is not there any more. So, yeah, it tries [...], but it doesn't work. (Guitarist 2, extract of post improvisation interview, scenario1_3-02, pers. comm.)

One last type of constraint is due to the acoustic and electric behaviour of the piezoelectric hexaphonic pickup. Indeed crosstalk [30] and the acoustic transfer of the strings vibrations through the bridge create resonances on non-played strings. These two phenomena are particularly significant in scenario 1, where notes played on strings without effects would trigger low volume modified sounds on strings with effects. This is especially true when the distortion is used on the string with effects as it significantly increases the volume of the strings. However, these constraints were used by guitarist 2, 3 and 4 as a mode of playing in itself.

7.2 Appropriation of the System

In spite of the constraints brought by the scenarios and by the technical system, the guitarists have, for the great majority, succeeded in appropriating and integrating the complexity of the system. They have, for the most part, put in place various strategies to reduce this complexity.

[27] The *scordatura* is the action of strongly detuning the strings in order to be able to play with the timbres generated in part by the softness of the relaxed strings.

[28] `https://www.sonicvisualiser.org/` (08/02/2022).

[29] `https://www.advene.org/` (08/02/2022).

[30] A small amount of the sound of a vibrating string is picked up by adjacent pickups.

Guitarist 3, for example, played several improvisations using only a limited number of prepared strings [31] (scenario 1_2) or a limited way to attack the strings (scenarios 2_1, 2_2, 2_3). In scenario 1_2, guitarist 4 used preparation (a small bar of metal inserted in between the strings close to the bridge) [32] on the three dry strings in order to bring their timbre closer to the ones of the strings with effects. The same guitarist used *scordatura* right from the beginning (scenario 1_1) to limit the harmonic range of the guitar [33] :

> It appeared to me as a way, in this chaos of the low strings [strings with effects, editor's note], to try to find an organization clearer for me. (Guitarist 4, extract of post improvisation interview, scenario1_1, pers. comm.)

In this improvisation, the standard tuning of the guitar, E-A-D-G-B-e, evolved to E-A-D-A-A-e (the D string was left unused during all the improvisation).

During scenario's test step, guitarist 5 played pieces of his repertoire while applying effects in a random manner. Both of these actions (known pieces playing and randomness) helped him to dig directly into hexaphonic effects timbre without overthinking too much. From there he narrowed down to specific choice of effects and presets as well as specific directions for his improvisations (most of the time inspired by the pieces he just played). It has to be noticed that this guitarist, despite having found ways to work with the setup, didn't finish the experiment as he felt stuck. For this specific guitarist, more time would have been needed to interact and play with the system.

7.3 Guitarist Practice

The playing of an electric guitar is necessarily modified by the addition of an hexaphonic pickup and of a string-individualized sound processing system. The gestures made by the guitarists no longer have the same scope, the same impact, which leads him to reexamine his relationship with his instrument and therefore its practice. However hexaphonic multieffect was not considered by the guitarists as something radically new. As guitarist 2 stated: "It's another instrument, but at the same time, it's a familiar one" [34] .

Guitarist 1 develops the same idea :

> [...] because there's also the habit of linking instrumental gesture with foot gestures on the pedals. So there you have it, it seems to me to fit more into the same logic, but at a higher level we'll say. (Guitarist 1, additional interview, pers.comm.)

With this quote, guitarist 1 emphasizes that working with the foot controller in scenario 2 is pretty close to its practice of monophonic guitar effect pedals but it adds more relief or details to it. This idea of being part of a known and familiar practice which goes further and needs to adapt is developed by guitarist 2 :

> It doesn't question the practice. The practice stays, it exists. However, it reconsiders it, in the sense where, as new things happen, you have to adapt. [...] It's a system that forces you to look deeply at the instrument for ways to adapt, even if it is a material that I know. I mean, all these effects, I've already used them in my life. They are part of the guitarist's landscape. And despite all that, the fact to use them in hexaphony, you rethink the effects differently too. You rethink, you adapt your playing, an interaction takes place. (Guitarist 2, extract of post improvisation interview, scenario2_5, pers. comm.)

The same guitarist sums up this idea in a pretty concise and concrete way when he mentions, "With this system, you have to redraw the geographical relationship with your instrument" [35] . This "geographical relationship" is close to the notion of "mapping" often used in NIME (New Interface for Musical Expression) context. Using an hexaphonic multieffect pushes the guitarist to modify the relationship, one could say the "inner mapping", he has built between its gesture and the resulting sound.

From these few excerpts, it appears that this hexaphonic setup can easily integrate into practices of guitarists as it uses the same elements as their regular practices, but guitarists also need to adapt their practices in terms of the relationship between the gesture and the resulting sound.

7.4 Hexaphonic Specificities

Regarding the produced sounds, several comments develop the idea of this setup having a rich palette of sounds. Guitarists 1 and 4 both make the parallel with the organ. This metaphor derives directly from the use of the foot controller in the second scenario but also from the vast number of sounds accessible through this controller and through the playing. Let's remember that scenario 2_5 gives access to individual bypass controls of individual effects as well as to presets of the whole bypass matrix. These options give the musicians the ability to change configurations completely or just to make tiny adjustments, these abilities to be modulated by guitarist's playing which itself acts as a "selection gesture" [21] inside the defined timbre palette. Guitarist 2 develops this idea of rich sound possibilities by comparing those to the ones accessible through Digital Audio Workstations (DAWs) software :

> Precisely, it helps going into fields that could be covered by the digital world and all that. I find that we come close to things like that, while respecting the instrumental practice. [...] I'm not saying that it's like MIDI [...], it's a kind of in between. (Guitarist 2, extract of

[31] Instrument's preparation is the process by which object are added or fixed on the strings in order to modify its original timbre.

[32] See video recording at `https://vimeo.com/639069813` (08/02/2022).

[33] See video recordings at `https://vimeo.com/639069333` (08/02/2022).

[34] Guitarist 2, extract of post improvisation interview, scenario 2_5, pers.comm

[35] Additional interview, Guitarist 1, pers. comm.

> post improvisation interview, scenario2_5, pers. comm.)

"Respect of instrumental practice" here, comes in opposition to MIDI control of instruments. With the hexaphonic system, the guitarist uses the professional practice he spent years developing (and not a reduction of it) accessing all these sound possibilities.

The specificities of the practices used during this experiment are also the centre of the last hexaphonic specificity developed here. Indeed, it appears that practices that already made use of techniques to develop polyphonic or multitimbral approaches of the guitar were enhanced by this system. Guitarist 3 who extensively used preparations during its improvisation points out at several moments the gain of clarity due to the hexaphony :

> [...] one can add a tremolo or a delay on just one string, it's really nice. Or a reverb on one string brings a bit of depth and it won't impact everything. On a classic analogue effect, it necessarily takes a huge proportion. Here, one can add a huge reverb, but just on one string, it's very convenient yeah. (Guitarist 4, extract of post improvisation interview, scenario2_4-01, pers. comm.)

Despite being obvious when said aloud, this remark brings to the fore the idea that monophonic effect pedals are not that well suited for prepared guitar practice. Indeed, added preparations can be string-specific (as well as applied to multiple strings) whereas monophonic effects do apply to all the strings at the same time. In such a context, string-specific effect system (aka hexaphonic multieffect) brings a natural continuity to preparations. Guitarist 4 whose practice falls into classical, contemporary and flamenco styles tends to formulate the same kind of idea:

> With a classical guitar, we work with the aim of being able to have an action as independent as possible from each finger and therefore potentially also differentiated regarding the strings. (Guitarist 4, complementary interview, pers. comm.)

This quote comes as a justification of the idea brings by guitarist 4 that the hexaphonic multieffect setup could be of interest of classical guitarists who would want to move on to electric guitar. As he mentions, the independence of the finger from each other seems to find a continuity in a string-specific audio effects system. This point (which wasn't expected in the first place) and the example of the prepared guitar practice seems to highlight that this kind of system is a good fit for practices that seeks independence in terms of gestures, of strings or of preparations.

8. FUTURE WORK

While bringing some important notions, the analysis presented above remains a first step. As mentioned above the small number of guitarists limits the generalization of the findings. More guitarists and a more balanced dataset would definitely help improve the results. More amateur guitarists evolving in rock-related style of music may have helped understand better why guitarist five felt stuck. A wider variety in the music styles would also help as styles often come with specific modes of playing. Those specific modes of playing can be not represented in our dataset,and could, more than probably, adapt differently to hexaphonic effects.

Another area that should profit from future work is the visualization part. Indeed, we showed a data visualization test in Advene software but, while being able to show all the data (except audio signals) at once, this software is not optimum to run a study with such a number of different types of data. Being able to switch between audio signals (clean or wet hexaphonic, monophonic or sound from video) while looking at the same video or to summarize relevant data for specific parts of an improvisation are some of the features which would be really useful to music analysis study.

Lastly, the first insights gained by the analysis of the interviews could be broadened by the addition of the objective data and their analysis. The use of a fretboard patterns theoretical analysis tool such as the model developed in [22] could help in finding a common ground to the different practices analysis. This common ground appears to be vital to conduct a more efficient comparison between guitarists' appropriation of the system. Further, in order to really understand the impact of the hexaphonic setup on the practice of a specific guitarist, an analysis of its regular practice (i.e. without hexaphonic setup) should be done so as to enable a "before/after" comparison.

9. CONCLUSION

This paper presents a new dataset including improvisations of five different guitarists using hexaphonic guitar and multieffect. Such a system enables them to apply string-independent effects. This dataset gathers 10 hours of improvisations and transcription of 4.5 hours of interviews. Subscenario's improvisation is annotated with pitch, string, fret, playing techniques (including extended techniques), effects configuration and effects activation timings. As these annotations represent a large amount of data and as the annotation process is not finished at the time of the writing, the first analysis given in this paper is based on the different interviews. According to the guitarists, this system, while new, appears to be familiar: effects are a well-known paradigm for electric guitarists and there's no need to learn new gestures; the guitarists' practices can be used as it is. What requires a new approach from the guitarists, though, is the impact the instrumental gestures have on the generated sound, what one could call "inner mapping". Indeed by being string-specific, hexaphonic effects add another level to the impact the gestures can have. As a wrap-up, guitarists make several remarks that tends to emphasize that this system is particularly relevant with practices that already use the independence of fingers and strings extensively, such as prepared or classical guitar practices.

Acknowledgments

I'd like to thank Livio Ratti, student at the University of Lille (France) for his precious help during the recording sessions and the video editing steps. This work was partially funded by the Walloon Region and the EU, in the framework of ERDF project DigiSTORM.

10. REFERENCES

[1] J. Thickstun, Z. Harchaoui, and S. M. Kakade, "Learning features of music from scratch," in *5th International Conference on Learning Representations ICLR*, 2017.

[2] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.

[3] C. Hawthorne, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: dual-objective piano transcription," in *Proceedings of the International Symposium on Multimedia Information Retrieval, 2018*, 2018, pp. 50–57.

[4] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score- and Instrument-Related Parameters," *Proc. of the 17th Int. Conference on Digital Audio Effects (DAFx-14)*, pp. 1–8, 2014.

[5] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "Guitarset: A dataset for guitar transcription," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, E. Gómez, X. Hu, E. Humphrey, and E. Benetos, Eds., 2018, pp. 453–460.

[6] L. Su, L.-F. Yu, and Y.-H. Yang, "Sparse cepstral and phase codes for guitar playing technique classification," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 9–14.

[7] C.-R. Nadar, J. Abeßer, and S. Grollmisch, "Towards cnn-based acoustic modeling of seventh chords for automatic chord recognition," in *Proceedings of the 16th International Conference on Sound And Music Computing (SMC)*, 2019.

[8] M. Stein, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic Detection of Audio Effects in Guitar and Bass Recordings," in *Proceedings of the 128th Audio Engineering Society (AES) Convention*, 2010.

[9] R. Foulon, P. Roy, and F. Pachet, "Automatic Classification of Guitar Playing Modes," in *Sound, Music, and Motion : 10th International Symposium, CMMR 2013*, M. Aramaki, O. Derrien, R. Kronland-Martinet, and S. Ystad, Eds., vol. 8905. Marseille, France: Springer, 2014, pp. 58–71.

[10] P. D. O'Grady and S. T. Rickard, "Automatic Hexaphonic Guitar Transcription Using Non-Negative Constraints," in *Proceedings of the Irish Signal and Systems Conference*, 2009.

[11] M. Pfleiderer, K. Frieler, J. Abeßer, W.-G. Zaddach, and B. Burkhart, Eds., *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017.

[12] M. Puckette, "Patch for guitar," in *Proceeddings, PD Convention, 2007*, 2007.

[13] L. Reboursière, "Traitement sonore polyphonique et contrôle gestuel instrumental : retour sur une mise en oeuvre pratique de la guitare hexaphonique (to be published)," in *Proceeding of the symposium "When the guitar electrifies !"*, ser. Collection MusiqueS, B. Navarret, M. Battier, P. Bruguiere, and P. Gonin, Eds., vol. MusiqueS & Sciences. Sorbonne Université Presses, 2020.

[14] R. Graham, "Expansion of electric guitar performance: Pratice throught the application and development of interactive digital music system," Ph.D. dissertation, Faculty of Creative Arts, University of Ulster, 2012.

[15] E. Bates, "The Composition and Performance of Spatial Music," Ph.D. dissertation, University of Dublin, 2009.

[16] P. Brossier, "Automatic annotation of musical audio for interactive systems," Ph.D. dissertation, Centre for Digital music, Queen Mary University of London, 2006.

[17] S. F. Josel and M. Tsao, *The Techniques of Guitar Playing*, C. Nobach, Ed. Barentreiter, 2014.

[18] J. Schneider, *The Contemporary Guitar*. Roman and Littlefield, 2015, ch. 7 - Microtones: The Well-Tuned Guitar, pp. 141–214.

[19] B. Hopkin and Y. Landman, *Nice Noise : Modifications and Preparations for Guitar*. Experimental Musical Instruments, 2012.

[20] M. Elgart and P. Yates, *Prepared Guitar Techniques*. California Guitar Archives, 1990.

[21] C. Cadoz, "Le geste canal de communication homme/machine - la communication "instrumentale"," in *Revue des Sciences et Technologies de l'Information - Série TSI : Technique et Science Informatiques*. Lavoisier, 1994, vol. 13, no. 1, pp. 31–61.

[22] J. De Souza, "Fretboard Transformations," *Journal of Music theory*, vol. 62, no. 1, pp. 1–39, 2018.

SMC-22 *JIM Jeune Chercheur* Papers
(in French)

SOUTENIR EN CLASSE L'ÉCOUTE ACTIVE, L'AUTONOMIE ET L'ÉCHANGE EN ANALYSE MUSICALE AVEC LA PLATEFORME WEB DEZRANN

Alice Sauda
Université Paris 8, EA 1572 Musidanse, CICM
alice.sauda@etu.univ-paris8.fr

Mathieu Giraud *Emmanuel Leguy*
Univ. Lille, UMR 9189 CRIStAL CNRS
{mathieu,emmanuel}@algomus.fr

RÉSUMÉ

Avoir une écoute active, comparer des éléments musicaux et construire une culture musicale commune font partie des buts de l'éducation musicale. La diversification des supports d'enseignements et l'utilisation d'outils numériques sont largement recommandés. Nous présentons comment la plateforme libre d'analyse musicale Dezrann peut soutenir cette écoute active. Au cours de l'année 2021/22, 25 séances ont lieu dans 5 établissements de l'Académie de Lille, partiellement encadrées par une médiatrice et les enseignant·es d'éducation musicale. Des fiches réutilisables pour une première séance sont à disposition, et les enseignant·es peuvent aussi demander l'inclusion d'autres pièces. Nous discutons dans ce premier bilan des possibilités de ce type d'outil ainsi que de la prise progressive d'autonomie des élèves.

1. INTRODUCTION

Pédagogie de l'éducation musicale. La pédagogie recherche l'engagement des élèves, qu'il soit dans la classe ou qu'il se prolonge à l'extérieur [13]. En éducation musicale, une modalité particulière d'engagement est l'*écoute active*. En psychologie et communication orale, l'écoute active, développée à partir des travaux de Carl Rogers, demande de porter son attention à la fois sur le contenu mais aussi sur les intentions ou les ressenti du locuteur. En musique, l'écoute active demande à se concentrer sur ce qui « fait sens » dans la musique : les sons, la structure et d'autres les élèments qualitatifs, et potentiellement aussi se projeter dans l'univers de l'interprète voir du compositeur [14, 16, 20]. En plaçant l'élève comme « actif », une telle écoute favorise l'engagement [12], et le pousse à écouter de manière réfléchie tout en créant des liens émotionnels avec la musique [15].

Au cours d'un exercice d'écoute, l'attention des élèves sur la musique est très courte, d'où l'importance d'exercices qui concentrent l'attention sur de petites sections par un travail d'analyse musicale progressif et dirigé. Les élèves qui cherchent quelque chose de précis dans la musique « décrochent » moins [6].

En France, le programme d'éducation musicale de cycle 4 (classes de 5e, 4e et 3e) se structure autour de quatre grands champs de compétences établis à partir du socle commun, là encore visant à un engagement des élèves : Réaliser des projets musicaux d'interprétation ou de création ; Écouter, comparer, construire une culture musicale commune ; Explorer, imaginer, créer et produire ; Échanger, partager, argumenter et débattre [26]. Les enseignant·es réalisent des séquences autour de la production et de la perception sur des domaines musicaux complémentaires :

- Les domaines du *timbre* et de l'*espace*, du temps et du rythme et de la dynamique qui concernent les caractéristiques des matériaux sonores qui constituent le discours musical ;
- Le domaine du *successif* et du *simultané* qui renvoie à l'organisation temporelle, horizontale et verticale, du matériau musical (microstructure) ;
- Le domaine de la *forme* concerne l'organisation des matériaux et techniques sur le temps long de la musique (macrostructure) [24]

Ces domaines s'articulent autour de la notion de *style*, dans une démarche de comparaison entre les répertoires, les œuvres et les périodes de l'histoire de la musique.

Les enseignant·es diversifient les activités au cours de leurs séquences (typiquement 4 à 6 par an de chacune 4 à 7 heures de cours), et sont habitués à utiliser en alternance plusieurs supports ou médiations, en particulier pour l'écoute active. Pendant l'écoute, les élèves peuvent par exemple mobiliser leur *corps* et/ou leur *voix* (lever la main lors de l'identification d'un thème, chanter/dire des choses), mais aussi des supports *papier* ou *physiques* par la réalisation de musicogrammes ou d'annotations sur une ligne de temps.

Outils numériques pour la pédagogie musicale. Le numérique est présent en classe d'éducation musicale, déjà dans la possibilité de jouer un fichier son ou une vidéo, possiblement de l'annoter au tableau blanc interactif [25] ou bien avec des solutions comme ActivInspire, ou l'utilisation de logiciels musicaux comme Audacity. Des

Collège Michelet, Lens (62)	5e, option musique électronique	15 élèves, 12-13 ans
Collège Félix del Marle, Aulnoye-Aymeries (59)	5e et 4e, option musique	13 élèves, 12-14 ans
Collège Jean Rostand, Marquise (62)	4e	24 élèves, 13-14 ans
Collège Brossolette, Noyelles-sous-Lens (62)	3e	21 élèves, 14-15 ans
Lycée Voltaire, Wingles (62)	Seconde, option musique	16 élèves, 15-17 ans

Table 1. En 2021/22, environ 90 élèves de 5 classes de l'Académie de Lille participent au programme de médiation scientifique « Les sciences infusent » autour de Dezrann, y compris 2 élèves en situation de handicap intégrés en ULIS.

outils plus spécialisés pour la recherche et/ou la pédagogie se développent, qu'ils soient au service de la créativité, de la production musicale, de l'écoute ou de l'analyse [2], comme par exemple, l'Acousmographe [8] pour la visualisation et l'annotation, la plateforme SoloTutti pour la chorale à distance [23], l'environnement Kiwi pour la création musicale en temps réel [17], et son application pour des ateliers de *live patching* [18] ou encore la plateforme Mediate pour l'analyse audiovisuelle [3].

L'éducation nationale encourage largement l'utilisation de ces outils [23, 25]. Chaque académie a désormais, par matière, un Interlocuteur Académique Numérique (IAN). En réponse à la crise sanitaire, les outils numériques sont susceptibles de favoriser la continuité pédagogique entre un travail en classe et à distance [19], cela dépendant aussi des équipements et connexions des élèves.

Il n'existe pas de réel consensus sur la manière de mesurer l'engagement des élèves, et a fortiori l'impact des outils numériques en pédagogie : les études quantitatives sont délicates [5, 19]. En faisant une revue d'une vingtaine d'études sur le sujet, Venn et al. soulignent cependant que c'est la *manière dont ces outils sont utilisés*, en particulier l'encadrement des élèves, qui agit sur l'engagement [19]. Le choix du répertoire, même s'il ne peut jamais satisfaire toute une classe, a aussi une influence sur la motivation [12] et travailler en partie sur un répertoire populaire pourrait être plus attrayant [21], en particulier chez les jeunes adolescents [15].

La plateforme d'analyse musicale Dezrann. La plateforme web Dezrann [1] est l'un de ces outils pour favoriser l'écoute active, et plus généralement pour étudier et annoter la musique (partition et/ou forme d'onde) avec des *étiquettes* ponctuelles ou ayant une certaine durée (Figure 1) [10, 11]. Dezrann a été initialement conçue comme outil pour la recherche en informatique musicale, plus précisément en analyse musicale computationnelle (CMA, *computational music analysis*), et utilisée pour l'annotation de motifs, cadences et de structures dans les fugues et formes sonates [1, 9]).

En 2018/19, lors d'un premier projet avec l'académie d'Amiens, nous avions observé des scénarios collectifs d'écoute active et d'analyse (collège Amiral Lejeune à Amiens, et collège Gérard Philippe à Soissons) : une première écoute, une identification de vocabulaire, des écoutes ultérieures avec utilisation ou précision du vocabulaire pour décrire des éléments musicaux. Certain·es enseignant·es utilisaient déjà des outils numériques (annotations sur tableau blanc interactif, parfois sur forme d'onde ou outils en ligne), mais sans solution permettant de facilement annoter de la musique et de comparer et de sauvegarder ces annotations. Nous avions commencé à faire évoluer Dezrann pour répondre à ces scénarios en essayant d'avancer vers une autonomie des élèves, et effectué à Amiens des premières séances en 2019/20 [22].

Contenu. Nous décrivons dans cet article comment la plateforme web d'analyse musicale Dezrann peut contribuer à des séances d'éducation musicale visant à développer l'écoute active des élèves. Dans le cadre du programme de médiation « Les sciences infusent » [2] de l'Université de Lille, cinq établissements ont en 2021/22 des séances régulières (partie 2). L'objectif principal est de *donner envie d'écouter activement la musique*, de mettre des mots sur la musique. Les séances sont aussi le lieu de médiation scientifique, pour partager nos travaux académiques, et sont enfin l'occasion d'avoir des retours sur l'utilisation de Dezrann afin de l'améliorer. Nous dressons donc ici le bilan de plus de 15 séances en milieu scolaire et proposons des réflexions sur les réussites, limites et enjeux de tels outils pédagogiques (partie 3).

2. LES ATELIERS « LES SCIENCES INFUSENT » AUTOUR DE LA PLATEFORME DEZRANN

2.1. Organisation des ateliers

Les premières séances avec Dezrann ont débuté en 2019 dans l'académie d'Amiens. En 2021/2022, un programme pilote de l'Université de Lille, « Les sciences infusent », en collaboration avec l'académie de Lille, prévoit environ 25 séances d'une heure de classe dans cinq établissements, collèges et lycée (Table 1). Ces cinq établissements ont été choisis par l'Académie pour diversité d'âges, de situations socio-économiques, et la pré-

1 . http://www.dezrann.net

2 . https://sciencesinfusent.univ-lille.fr/

Figure 1. Une analyse structurelle de *Money* (Pink Floyd) réalisée avec Dezrann. Les étiquettes peuvent être créées à la volées ou éditées, y compris avec les *boutons raccourcis* que l'on peut définir dans la barre du bas.

Rondo de la *Suite Abdelazer*, Purcell	extrait du *Concerto pour violon op. 61* en Ré majeur, Beethoven	*Go down Moses*, L. Amstrong
Canon en Ré, Pachelbel	*Samuel Goldenberg und Schmuÿle* dans les *Tableaux d'une exposition*, Moussorgsky	*When the Saints*, L. Amstrong
Te Deum, Charpentier	*Éléphant* du *Canaval des animaux*, Saint-Saëns	*Cantaloupe Island*, H. Hancock
Fugue en Do mineur BWV847, Bach	*Roi des Aulnes*, Schubert	*Stand by me*, Ben E. King
Les Sauvages dans les *Indes galantes*, Rameau	extraits de l'*Arlésienne*, Bizet	*Money*, Pink Floyd
extraits des *Quatre saisons*, Vivaldi	*Dans les steppes de l'Asie centrale*, Borodine	*Le Soldat*, F. Pagny
La Campanella, Paganini	*Valse n°2*, Chostakovitch	*Gloria* de la *Misa Criolla*, A. Ramirez
extrait de *Water music*, Haendel	*Les entretiens de la belle et la bête*, Ravel	extrait de *Don Quixotte Corp.*, Savoret
premier mouvement du quatuor *Op. 33 n° 2*, Haydn	*Boléro*, Ravel, et adaptation d'A. Kidjo	*Chants de pygmées*
variations sur *Ah vous dirai-je maman* K265, Mozart		*Danse de l'âme*, trad. Maroc
		Kahwa shemesh, F. Atlan, M. Oudwan

Table 2. Les pièces étudiées en séance couvrent une variété de styles musicaux. Les pièces peuvent être ajoutées à partir de fichiers audios (fichier son, vidéo ou lien YouTube) et/ou représentés symboliquement (fichier MIDI, MusicXML, MEI), et il est possible de synchroniser ces représentations. Ces pièces sont principalement de la musique tonale et quelques musiques du monde : le catalogue devrait être étendu, en particulier vers des musiques médiévales ou de la renaissance et vers des musiques (post)-sérielles, électroniques et/ou mixtes.

sence d'enseignant·es moteurs en utilisation d'outils numériques.

Chaque classe réalise entre 4 et 6 séances d'une heure avec la plateforme, avec leur enseignant·e, en partie aidée par une médiatrice et/ou un·e chercheur ou chercheuse. Tous ces adultes souhaitent partager un discours sur la musique (et sur l'informatique musicale !). La plupart des enfants n'ont aucune formation musicale spécialisée et sont "non-lecteurs" de partition. Pour ces enfants, un support visuel informatif sera la représentation de l'audio en *forme d'onde*, éventuellement complété par une vidéo, et exceptionnellement par une partition, qui peut être une aide visuelle même pour des non-lecteurs. Dezrann peut afficher ces trois supports.

Lors de la première séance, la médiatrice introduit une pièce choisie par l'enseignant·e parmi un catalogue, et utilise la plateforme pour écouter et annoter des éléments musicaux (Figure 2). Les élèves se déplacent eux aussi à tour de rôle au poste central (suivant les lieux, ordinateur du bureau, tableau interactif avec stylet). Dezrann se veut aussi « transparent » que possible, afin de se focaliser sur les concepts musicaux sur lesquels les enfants travaillent plutôt que sur l'outil numérique en soi. Dans les séances suivantes, l'enseignant·e choisit librement n'importe quelle pièce. Qu'il y ait ou non présence de la médiatrice, ces séances visent à la fois à améliorer l'autonomie des enfants (seuls ou en groupes de 2-4) en écoute et analyse de la musique – mais aussi à permettre que leur travail soit partagé et discuté (Figure 3).

2.2. Pièces et notions musicales abordées

Que ces pièces soient déjà connues ou non par les élèves, une séance avec Dezrann met en œuvre les compétences souhaitées en éducation musicale. Le but collectif et/ou individuel est ainsi d'identifier et/ou de qualifier :

Figure 2. Séance 1 au collège Brossolette (Noyelles-sur-Lens, Pas de Calais), classe de 3e d'Eulalie Tison, analyse collective de *Money* avec la médiatrice.

— les instruments, les voix, ou plus généralement les sons et autres bruits, ainsi que leurs entrées (timbre, espace)

— des motifs, des thèmes, des similarités et contrastes, et d'aller ainsi vers une structuration de la pièce (successif, simultané, temps, dynamique, micro- et macro- structure)

Pour cela, un utilisateur de la plateforme peut créer de nouvelles étiquettes, les positionner sur la forme d'onde (ou la partition), les modifier, et facilement naviguer dans la pièce pour réécouter une partie de celle-ci ou bien la partie étiquetée (Figure 1, et vidéo [3]).

Les enseignant·es demandent l'ajout des pièces de leur choix à la plateforme (Table 2). Les œuvres ou enregistements protégés ne sont accessibles qu'après identification, dans le cadre de l'exception pédagogique.

De plus, pour guider la médiation lors d'une première séance, nous avons réalisé, en lien avec les enseignant·es, des fiches prêtes à l'emploi et les diffusons sous licence libre [3] . Les premières fiches finalisées portent sur :

— *Money (Pink Floyd)* : identification et qualification des 7 sons de la boucle, introduction de la mesure 7/4, découverte progressive de la pièce, réflexion sur les rôles des instruments tels que *accompagnement* ou *solo* ;

— *Dans les steppes d'Asie Centrale (Borodine)*, identification d'éléments de l'argument d'une musique à programme (ici thème chant russe, thème oriental, son tenu à l'aigu pour le désert, contretemps aux cordes en pizzicato pour les pas des chevaux), réflexion sur la texture orchestrale.

3 . Les fiches, la vidéo et le code source sont disponibles à partir de le page http://www.algomus.fr/dezrann

3. RÉUSSITES ET ENJEUX DE CES ATELIERS

Au 15 avril 2022, 18 séances sur les 25 prévues ont été effectuées et un premier bilan a été fait avec les enseignant·es et les inspecteurs et inspectrices d'académie.

Les objectifs pédagogiques de prise progressive d'autonomie ont été atteints dans toutes les classes. Pour les séances 1 en médiation, les classes ont été très volontaires et motivées à faire progresser les analyses. L'intervention d'une tierce personne dans leur classe ainsi que l'utilisation d'un nouvel outil numérique a attisé la curiosité des élèves. Les manipulations collectives de la séance 1 voire de la séance 2 favorisent l'autonomie dans l'utilisation de Dezrann pour les séances suivantes. Les élèves travaillent en autonomie sans difficulté seuls ou en groupe à partir de la séance 2 ou 3. Pour *Money*, travailler sur le bruit a été très ludique pour les élèves, et l'analyse dirigée et concentrée sur de petites sections et éléments précis a largement favorisé la concentration.

Les prochains paragraphes discutent ainsi sur le *positionnement de Dezrann et des outils numériques* en classe (partie 3.1), la *prise d'autonomie* observée sur ces ateliers (3.2) tout comme des *défis* pour l'utilisation de tels outils numériques (3.3).

3.1. Utiliser Dezrann pour soutenir l'écoute en classe

Varier les outils et les situations d'apprentissage et d'appropriation de la musique est au cœur de la pédagogie en éducation musicale. Afin de s'adapter au public et aux œuvres, la diversification des supports d'enseignements et l'utilisation d'outils numériques sont largement recommandés par les programmes d'éducation musicale de cycle 4 (voir l'introduction).

Les outils d'analyse et d'annotation comme Dezrann permettent de réaliser des exercices de comparaisons pour développer les compétences liées au domaine des représentations du monde et l'activité humaine [26]. La manipulation de divers outils ainsi que la comparaison de plusieurs versions d'une même œuvre permet d'explorer avec les élèves ce qui fait les différences et similitudes pour aborder les questions de l'histoire et des répertoires liées au domaine du style à travers les domaines complémentaires (forme, timbre et espace, temps, rythme et dynamique, successif et simultané). Dezrann apporte ainsi la possibilité de visualiser toute la pièce, de comparer les éléments et amène de nouvelles discussions en classe sur la méthodologie d'analyse ainsi que sur ce que révèlent les différents éléments sur le discours musical.

Figure 3. Séance 3 à Wingles (Pas-de-Calais) en seconde, option musique, sur le *Te Deum* de Marc-Antoine Charpentier, classe d'Emanuele Battisti. Après avoir été en petits groupes pour travailler en autonomie, les élèves se rassemblent et partagent et discutent leurs analyses entre eux et avec leur professeur.

3.2. Analyses collectives et individuelles : vers une prise d'autonomie des élèves

Les fiches utilisées lors des séances collectives ont servi de point de départ à l'acquisition d'autonomie dans l'usage de Dezrann pour l'analyse musicale. En séances collectives, la première écoute sans support ni instruction permettait de relever les premières impressions sur l'œuvre. Les élèves ont toujours eu pour réflexe d'identifier les instruments et de qualifier le sentiment général provoqué par l'écoute (triste, joyeux…). La thématique de l'atelier n'étant pas connue, les élèves convoquent les notions de la séquence en cours pour "répondre correctement" en utilisant à la fois leurs compétences de communication du socle commun et le vocabulaire travaillé en éducation musicale. Face à un exercice sans consigne qui demande de faire appel à l'imagination, les élèvent peuvent être déstabilisés [4].

Caractéristique d'une *pensée convergente et flexible*, ce réflexe permet de répondre à une exigence ou une consigne par l'identification d'éléments musicaux précis comme la dynamique, la mélodie ou le sentiment général [12].

On observe cependant un changement de pensée lors du travail individuel. Les élèves, seul·es ou en petit groupe, sont moins inhibé·es, même les plus timides qui explorent de nouvelles possibilités créatives. Le vocabulaire des étiquettes, choisi avec leurs propres mots, est plus subjectif et centré sur leur perception et leur ressenti. Cette pensée qui mobilise les compétences du socle commun lié à l'expression du langage des arts et du corps, s'inscrit davantage dans une *pensée divergente*, à la fois fluide et flexible.

Si l'écoute musicale implique une pensée divergente et convergente, la pensée divergente peut être développée par des exercices d'écoute active. Par exemple, par comparaison de parties contrastantes et l'identification de changements dans la musique [12], et donc par un travail d'analyse.

La pensée divergente adoptée naturellement révèle une capacité des élèves à s'adapter mentalement à l'exercice en fonction du contexte. Elle est plus créative car elle amène à trouver plusieurs réponses ou idées en faisant appel à l'imagination. La pensée fluide lors des activités individuelles démontre qu'ils se concentrent plus sur ce qui les intéresse dans la musique. Cette pensée fluide est-elle encouragée par le travail individuel ? Dans quelle mesure l'outil utilisé favorise la pensée divergente chez l'élève ? Un travail individuel et moins dirigé pourrait donner lieu à une attitude plus naturelle dans l'exercice tout en conservant l'engagement de l'élève [6].

Dans tous les cas, la prise d'autonomie entre les séances collectives et les séances individuelles semble avoir favorisé la créativité et l'autonomie de l'élève aussi pour adopter la méthode d'analyse qui lui convient le mieux. Certains élèves écoutent avant d'annoter et utilisent le logiciel au même titre qu'un support papier, tandis que d'autres vont explorer les indications offertes par l'onde au fur et à mesure de l'écoute.

3.3. Quelques défis de l'utilisation d'outils numériques

Le travail demandé doit être réaliste au vu des contraintes de temps. Une « heure de cours », à plus forte raison avec installation en salle pupitre, revient au mieux à 40-45 minutes de temps effectif de concentration. Ce temps peut s'améliorer au cours de séances successives. Même en diversifiant les activités, il est recommandé de se focaliser

sur un concept ou deux suivant la séance. Typiquement, sur *Money*, nous avons pu améliorer la fiche au bout de plusieurs séances dans différents établissements. Le travail d'une séance se focalise sur l'introduction, et la structure d'ensemble n'est qu'évoquée, on cherche surtout à réidentifier la boucle de sons. Cette formule permettait à la classe de découvrir progressivement le morceau et d'aborder d'autres notions musicales comme les techniques de jeu.

L'introduction de tout support autre que la musique (même non numérique, comme le papier ou le corps) est bénéfique, mais peut aussi divertir. Des éléments paramusicaux sont aussi très présents : dans les *Steppes de l'Asie Centrale*, le programme narratif attire l'attention des élèves plus que les éléments du discours musical, et les élèves cherchent dans ce cas à utiliser l'outil numérique d'abord pour souligner ce programme narratif... mais pourquoi pas, c'est justement de la musique à programme ! Ces digressions sont bénéfiques, l'enjeu d'une telle séance et du travail de l'enseignant·e est de les exploiter pour enrichir la discussion sur le contenu musical.

Développer les compétences numériques fait partie du socle. Cependant, la distraction générée par les outils numériques est aussi un défi pour les enseignant·es [16]. Enfin, certaines utilisations de Dezrann ne sont pas optimales comme l'annotation d'éléments trop rapprochés. Suite aux retour des élèves et des enseignant·es, nous avons adapté la plateforme au fur à mesure des séances.

4. CONCLUSIONS ET PERSPECTIVES

La pédagogie recherche l'engagement des élèves [13]. En éducation musicale, cet engagement peut notamment se rechercher dans l'écoute active pour explorer et décrire la musique. Bien utilisés, des outils numériques peuvent aider à cet engagement [19]. La plateforme Dezrann soutient ainsi des activités d'analyse musicale en classe, collectives et individuelles, en développant l'autonomie et l'engagement des élèves. Nous proposons des fiches pour débuter une telle activité et un support aux collègues enseignant·es pour l'ajout de nouvelles pièces et l'utilisation de la plateforme, ainsi que des éléments de médiation sur des recherches en informatique musicale. Il serait aussi intéressant d'avoir une évaluation plus précise, voire quantitative, de l'efficacité de ces séances, mais, outre les contraintes logistiques, un tel projet se heurterait aux difficultés d'évaluation de l'engagement des élèves [5, 19].

Lors de discussions avec les enseignant·es, nous avons vu que les usages de la plateforme peuvent être encore étendus, que ce soit pour des devoirs à la maison ou du travail à distance en continuité pédagogique. Dezrann peut aussi être un support de présentation et d'évaluation, que ce soit en travail en classe, à la maison, ou même parmi les supports de l'oral du brevet. De plus, durant les séances, des élèves ont eux-mêmes proposé de nouveaux usages. Une idée qui sera prochainement mise en œuvre est que chaque élève choisisse une pièce à analyser et la présente à la classe avec son analyse. La sélection du répertoire peut ainsi encore améliorer l'engagement de l'élève [12].

Des évolutions techniques de la plateforme sont enfin envisagées, notamment pour donner accès à un module de synchronisation améliorant l'entrée de pièces [10] et rendant possible de nouvelles activités pédagogiques.

Remerciements

Nous remercions fortement les enseignant·es et les inspecteurs et inspectrices d'académie (IA-IPR, *) des académies de Lille (Eulalie Tison, Rémi Gravelin, Guillaume Girin, Murielle Chaignier, Emanuele Battisti, Marie Gandin*, Laurent Raymond*) et d'Amiens (François Degroote, Vincent Louette, Anne-Isabelle Ramanantsitohaina*) pour leur implication enthousiaste dans ce projet, les élèves de ces classes, ainsi que Pauline Leroy et Camille De Visscher et l'Université de Lille pour la structuration et le suivi du programme « Les sciences infusent ».

Nous remercions également le CPER MAuVE et Louis Garczynski pour les développements sur Dezrann, le Mésocentre de Lille pour l'accès aux ressources de calcul et à l'hébergement, et l'ensemble de l'équipe Algomus pour les remontées sur la plateforme ainsi que pour les éléments de recherche présentés dans les supports de médiation.

5. RÉFÉRENCES

[1] Pierre Allegraud et al., *Learning Sonata Form Structure on Mozart's String Quartets*, Transactions of the International Society for Music Information Retrieval, 2(1) :82–96, 2019.

[2] Federico Avanzini, Adriano Baratè, Luca A. Ludovico, Marcella Mandanici, « A multidimensional taxonomy of digital learning materials for music education », dans *Pedagogies of Digital Learning in Higher Education*, 88–103, 2020

[3] Joel Burges, Solvegia Armoskaite, Tiamat Fox, Darren Mueller, Joshua Romphf, Emily Sherwood, Madeline Ullrich, *Audiovisualities out of Annotation : Three Case Studies in Teaching Digital Annotation with Mediate*, Digital Humanities Quarterly, volume 15(1), 2021

[4] Anaëlle Camarda, Mathieu Cassotti, « La créativité », dans *Neurosciences cognitives développementales*, De Boeck, 252-280, 2020

[5] Jean-François Céci, « Analyse de l'efficacité d'un dispositif de pédagogie active avec le numérique », dans *Vers de nouveaux modèles d'apprentissage,*

de pratiques pédagogiques innovantes et TIC pour l'éducation au développement durable, 2016

[6] Frank M. Diaz, *Listening and Musical Engagement : An Exploration of the Effects of Different Listening Strategies on Attention, Emotion, and Peak Affective Experiences*, Applications of Research in Music Education, 33(2), 27–33, 2015

[7] Zhiyao Duan, Slim Essid, Cynthia C.S. Liem, Gaël Richard, Gaurav Sharma, *Audiovisual Analysis of Music Performances : Overview of an Emerging Field*, IEEE Signal Processing Magazine, volume 36, 63–73, 2019

[8] Emmanuel Favreau, Yann Geslin, Adrien Lefèvre, *L'Acousmographe 3*, Journées d'Informatique Musicale (JIM 2010), 2010

[9] Mathieu Giraud, Richard Groult, Emmanuel Leguy, Florence Levé, *Computational fugue analysis*, Computer Music Journal, 39(2), 2015

[10] Louis Garczynksi, Mathieu Giraud, Emmanuel Leguy, Philippe Rigaux, *Modeling and Editing Cross-Modal Synchronization on a Label Web Canvas*, Music Encoding Conference (MEC 2022), 2022, *à paraître*

[11] Mathieu Giraud, Richard Groult, Emmanuel Leguy, *Dezrann, a web framework to share music analysis*, International Conference on Technologies for Music Notation and Representation (TENOR 2018), 104–110, 2018

[12] Teresa K. Hargrove, *Listening in the Secondary Music Classroom*, mémoire de master, Minot State University, 2019

[13] Robert-Vincent Joule, « De la théorie de l'engagement à la pédagogie de l'engagement ». *In*, Agnès Florin, Pierre Vrignaud (Eds.), *Réussir à l'école*, Presses universitaires de Rennes, 131-145, 2007

[14] Afsin Kémâl, *Psychopédagogie de l'écoute musicale, écouter, entendre, comprendre*, De Boeck, 2009

[15] Geoffrey M. Lowe, *Hearing but not Listening : Actively Engaging Students in Listening to Music beyond the Superficial in the Music Classroom*, International Yearbook for Research in Arts Education, volume 3, 2015

[16] Dan Mamlok, *Active Listening, Music Education, and Society*, Oxford Research Encyclopedia of Education, 2017

[17] Eliott Paris, Jean Millot, Pierre Guillot, Alain Bonardi, Anne Sèdes, *Kiwi : Vers un environnement de création musicale temps-réel*, Journées d'Informatique Musicale (JIM 2017), 2017

[18] João Svidzinski, Messina Marcello. *Live patching collaboratif – Vers une médiation inclusive avec l'informatique musicale*, Journées d'Informatique Musicale (JIM 2021), 2021

[19] Edward Venn, Jaeuk Park, LinePalle Andersen, Momna Hejmadi, *How do learning technologies impact on undergraduates' emotional and cognitive engagement with their learning ?*, Teaching in Higher Education, 1–18, 2020

[20] John L. Vitale, *The Importance of Active Listening and Reflection in the Music Classroom*, The Canadian Music Educator, 62(2), 38–42, 2021

[21] Ruth Wright, *Kicking the habitus : power, culture and pedagogy in the secondary school music curriculum*, Music Education Research, volume 10, 389–402, 2008

[22] Académie d'Amiens, *Favoriser l'écoute autonome en éducation musicale avec l'application Dezrann*, 2019

[23] Éducation Nationale, *Lettres ÉduNum éducation musicale*, https://eduscol.education.fr/2581/lettre-edunum-education-musicale

[24] Éducation Nationale, *Référentiels pour la construction des compétences : six domaines complémentaires*, https://eduscol.education.fr/document/18076/download, 2016

[25] Éducation Nationale, *L'enseignement de l'éducation musicale à l'heure du numérique*, https://eduscol.education.fr/document/18157/download, 2016

[26] Éducation Nationale, *Programme du cycle 4*, BOEN n°31 du 30 juillet 2020, https://eduscol.education.fr/document/621/download, 2020

PROTOTYPE DE GÉNÉRATION PROCÉDURALE DE CONTREPOINTS À LA MANIÈRE DE JEAN-SÉBASTIEN BACH

Jérémie Roux
Université de Montpellier (étudiant)
jeremie.roux@etu.umontpellier.fr

Violaine Prince
LIRMM - Équipe TEXTE
prince@lirmm.fr

RÉSUMÉ

L'objectif de cet article est de présenter un ensemble d'algorithmes permettant la génération automatique de fugues à la manière de Jean-Sébastien Bach. Il s'intéressera principalement aux différents constituants de la fugue : sujet, réponse, contre-sujet et épisode, pour lesquels des algorithmes de génération, définis selon les règles de *l'Art de la Fugue* [2], seront décrits et commentés. La séquence des constituants, fondée sur une analyse des fugues du *Clavier Bien Tempéré I* par l'équipe *Algomus*, est représentée sous la forme d'un graphe de transition (à la manière d'un modèle Markov caché) également détaillé. On s'intéressera également à la comparaison entre un modèle de transition des éléments des fugues selon la tradition de l'analyse musicale, et un autre issu de l'analyse statistique du corpus d'*Algomus*.

1. INTRODUCTION

La musique est un art que l'on peut qualifier de scientifique par sa construction, du fait de l'organisation décidée par le compositeur de l'enchaînement des rythmes et des notes, correspondant ou pas au style d'écriture de son époque. Certains compositeurs comme *Johann Sebastian Bach* (1685-1750) ont poussé cette conception de la musique à un point que l'on considère toujours aujourd'hui comme sortant de l'ordinaire. En particulier, lorsque l'on analyse *L'Art de la Fugue* [2], on ne peut s'empêcher de penser qu'il existe un certain nombre d'algorithmes implicites qui sont employés par le compositeur pour réaliser ses compositions (certains y voyaient une allusion aux mathématiques les plus "strictes" [4]). Cette régularité remarquable a conduit les informaticiens que nous sommes à nous poser deux questions.

Premièrement, est-il possible de transformer les algorithmes musicaux de Bach en algorithmes "informatiques", c'est-à-dire de les formaliser et de les transposer dans des ordinateurs, alors que la théorie musicale sous-jacente est éminemment complexe ? Et si oui, quel serait le prix de cette transformation, dans le sens de la simplification, de l'abandon éventuel de certains aspects, étant données les contraintes informatiques de la gestion de la complexité ?

Deuxièmement, si une telle action était possible, elle le serait sous forme d'un premier prototype destiné à évoluer incrémentalement avec chaque "victoire" sur la complexité. Mais elle aurait l'avantage de proposer des choix algorithmiques dans la continuité d'une mélodie donnée. Les oeuvres de *L'Art de la Fugue* sont des configurations particulières qui ont fixé les choix effectués par Bach au moment où il les a écrites. Ne pourrait-on pas simuler automatiquement, grâce à ce prototype, d'autres configurations qui réaliseraient, pour chaque étape, d'autres choix, et du coup, fournir de la matière première intéressante pour une analyse plus poussée des oeuvres en question ? Ce serait alors un usage musicologique inédit, un peu à la manière de ce qu'apportent des simulations informatiques à d'autres disciplines, en explorant l'arbre des possibles.

Dans le travail entrepris ici, nous nous sommes donnés pour objectif de réaliser une étude de type "preuve de concept", sachant que de précédentes avancées sur la modélisation pour la composition ont été largement passées au crible dans des travaux comme [11], s'appuyant entre autres sur le traité fondateur de Barbaud [12]. Pour cela, nous avons développé des algorithmes afin de théoriser et produire des morceaux sur la base d'une mélodie donnée en utilisant spécifiquement l'écriture contrapuntique de Bach dans [2]. Ces fameux algorithmes musicaux sont une *expression de l'écriture contrapuntique,* et celle de Bach est riche tout en obéissant à des règles strictes. Le contrepoint selon Bach est qualifié de *luxuriant* par Bougeret [3], qui l'oppose à l'écriture contrapuntique relatée dans le traité de Fux ([6]). Ce dernier s'appuie sur les pratiques non tonales de la fin de la Renaissance (Palestrina, Gesualdo) à laquelle furent formés les compositeurs de l'époque classique (tout en s'y opposant).

Le **contrepoint** est défini comme une forme d'écriture musicale qui trouve ses origines avec la polyphonie et qui consiste en la superposition organisée de lignes mélodiques distinctes (une définition fort didactique en est donnée dans [3]). Nous étudions les dimensions mélodiques, rythmiques et harmoniques de cette écriture, dont la **fugue** est le meilleur représentant.

Nous proposons ici un ensemble d'algorithmes de génération de constituants d'une fugue, selon notre inter-

prétation computationnelle de l'écriture de Bach, ainsi qu'une structure générale d'enchaînement des constituants, fondée sur un modèle de Markov caché [1] . Une ébauche musicologique de ce modèle est fournie puis complétée à partir d'une analyse automatique des fugues du *Clavier Bien Tempéré I* proposée par l'équipe *Algomus* des universités de Lille et d'Amiens [8], [7].

2. UN ÉTAT DE L'ART TRÈS SITUÉ

La musicologie a très longuement analysé les fugues et ce, tout au long des siècles. Le *Traité de la Fugue* de Gédalge [1] en donne une bonne définition, en particulier de la *fugue d'école*, qui est le principe à partir duquel on enseigne cet art aux étudiants en écriture. Les composants de la fugue auxquels nous nous référons dans les sections suivantes s'appuient sur des éléments issus de la pratique musicologique. Les travaux de Czaczkes [10] se focalisent davantage sur les fugues du *Clavier Bien Tempéré*, qui est un répertoire riche et sur lequel nous nous appuierons puisqu'il est au coeur de notre objectif (*i.e*, l'écriture contrapuntique de Bach), et aussi parce qu'une analyse computationnelle en a été faite et dont nous parlerons longuement.

En ce qui concerne l'informatique musicale et la génération de fugues, les travaux de Jan Philipp Hakenberg et Mathias Müller, auteurs de *The Pirate Fugue* avaient un objectif très proche du nôtre, et rien moins qu'audacieux : écrire le troisième livre (et non écrit par Bach) du *Clavier Bien Tempéré* ! La méthodologie en était cependant très différente : "*Rather than relying on encoding music theory into an algorithm, our approach is data driven : using a collection of over 5000+ digital classical scores...*" [2] . Les auteurs du logiciel se réfèrent également aux travaux de David Cope qui utilise des ATN (*augmented transition networks*) et une approche de type "informatique linguistique" pour générer automatiquement des compositions "dans le style" d'un compositeur ou un autre (dont des chorals de Bach) [5].

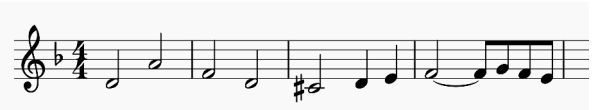
Les travaux qui nous ont personnellement le plus directement inspirés et ceux sur lesquels nous nous sommes appuyés sont les suivants : en analyse, les travaux de Giraud *et al.* se sont centrés depuis 2012 sur l'analyse computationnelle des fugues de Bach [8] en vue de la détection automatisée de motifs récurrents tels que *sujets* et *contre-sujets* (définis dans la section suivante) ainsi qu'*épisodes* pour en arriver à la définition d'un environnement complet d'analyse partagée [7] permettant à tout chercheur intéressé de pouvoir utiliser ces ressources en apprentissage comme en modélisation. En génération, outre ceux de Cope, les travaux les plus proches de notre problématique sont ceux de Hadjeres *et al.* en 2017 [9], qui ont proposé un système d'apprentissage automatique, fondé sur des réseaux de neurones, nommé *DeepBach*, entraîné par un apprentissage préalable sur des cantates de J. S. Bach. Il a été développé pour harmoniser automatiquement des thèmes musicaux dans le style du "choral", sachant que certains chorals comprennent des éléments canoniques, qui sont une sorte d'état initial des fugues.

Notre travail se trouve donc à la croisée des deux démarches : sur le même matériau que les travaux d'*Algomus*, nous avons construit un premier prototype de génération de fugue, ainsi complémentaire de l'analyse réalisée et fondé sur ses résultats pour alimenter notre structure d'enchaînement. Avec une démarche ayant le même objectif que *DeepBach*, mais aussi que [5] et *The Pirat Fugue*, nous nous sommes intéressés à la génération de fugues en optant pour un graphe basé sur la structure d'un modèle de Markov pour représenter les transitions entre différents éléments constituants des fugues, transitions dont les valeurs (ou poids) sont issues des travaux d'analyse des fugues [8] et en écrivant localement des algorithmes de génération pour chaque constituant. En ce sens, les travaux de Cope furent quelque peu précurseurs puisque l'ATN est aujourd'hui remplacé par les modèles de Markov dans les disciplines qui les utilisent.

3. ALGORITHMES ET ÉCRITURE CONTRAPUNTIQUE

Nous définissons et modélisons ici les différents éléments de l'écriture des fugues. Pour des raisons de simplification des algorithmes, nous nous en tiendrons à des fugues à 2 voix.

3.1. Sujet d'une fugue

Le **sujet** d'une fugue est le thème, motif mélodique et rythmique principal du morceau qui sera répété à l'identique, avec des variations et/ou des transpositions tout au long de l'œuvre. On donne donc en entrée du programme le sujet de la fugue que l'on veut obtenir ainsi que la tonalité de ce sujet (**Figure 1**).

Légende : b = blanche, n = noire, c = croche, $+$ = dièse, i = bémol

```
1 D m
2 D4_b A4_b F4_b D4_b C+4_b D4_n E4_n F4_2.5 G4_c F4_c E4_c
```

Figure 1: Sujet du premier contrepoint de L'Art De La Fugue (*BWV 1080*) en *Ré mineur* de J. S. Bach et fichier associé donné en entrée du programme

Le programme ne garde pas ce fichier tel quel, il crée une **voix** (tableau composé d'objets de type *note*) et chaque note de ce tableau correspond aux notes du sujet données en entrée du programme et conservées dans le même

1 . Lien vers le dépôt Git (code source et exemples) : *https://gitlab.etu.umontpellier.fr/jeremieroux/bach*

2 . *The Pirat Fugue : http://djtascha.de/the-pirate-fugues/*

ordre. Dans le codage du sujet présenté en **Figure 1**, les notes sont séparées par des espaces et l'information de la *hauteur* de la note est séparée par un '_' de l'information sur sa *longueur* (l'information de la longueur de la note dans un fichier donné en entrée du programme peut être codée par la première lettre du rythme correspondant mais aussi par un nombre de temps). Pour des raisons de conception, on ne peut donner que des sujets de fugue d'une longueur divisible par 4, c'est-à-dire qui remplissent complètement une suite de mesures de signature rythmique 4/4. On garde également en mémoire la tonalité du sujet qui est la tonalité principale du morceau.

3.2. La réponse, répétition transposée du sujet

La fugue démarre par une exposition du sujet en solo où l'on place le sujet donné en entrée sans le modifier sur une des voix disponibles. Il s'agit d'une voix plutôt grave si on veut qu'il soit répété à une hauteur plus aiguë, et inversement.

Tout de suite après l'exposition du sujet, il est répété à l'identique mais avec éventuellement une transposition (le plus souvent à la quarte ou la quinte, supérieure ou inférieure) sur une voix différente de celle sur laquelle il a été exposé : c'est la **réponse** [3] . Si le sujet est de longueur l, la réponse est donc de longueur l et la totalité de la longueur du morceau à l'instant où la réponse est terminée est de longueur $2l$ (**Figure 2**).

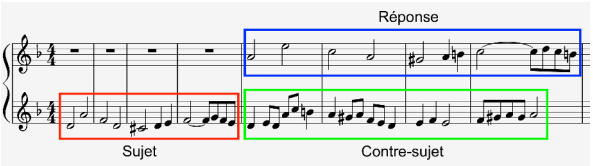


Figure 2: Réécriture du premier contrepoint de L'Art De La Fugue (*BWV 1080*) en *Ré mineur* de J. S. Bach avec le sujet original (la réponse et le contre-sujet sont générés par notre programme)

Algorithme 1 : EcrireReponse(S : sujet, t : nombre de demi-tons) : sujet transposé

pour chaque *note n* **de** S **faire**
  $n.hauteur \leftarrow n.hauteur + t$;
fin
renvoyer S;

Il n'y a donc pas de chevauchement entre le sujet et la réponse, la réponse commence sur le premier temps de la mesure d'après (et sur une autre voix) où le sujet termine. On implémente l'écriture de la réponse sur la voix 2 (la voix 1 étant par convention celle où le sujet est exposé en premier) comme décrit par l'**Algorithme 1**. On a bien la réponse qui commence après l'exposition du sujet, transposée de t demi-tons (variable qui peut être négative ou positive selon si la transposition est respectivement plus grave ou plus aiguë que l'original).

3 . Pour simplifier notre prototype, on effectue une réponse avec transposition exacte du sujet et non une réponse tonale avec mutation mélodique de la réponse (qui sert à conserver la même tonalité malgré le changement de degré de la réponse par rapport au sujet).

3.3. Le contre-sujet, accompagnement de la réponse

Le **contre-sujet** (**Figure 2**) est complémentaire mélodiquement et rythmiquement à la réponse. Il est sur la même voix que le sujet, de la même longueur et dans son prolongement, *i.e*, le passage du sujet au contre-sujet doit être imperceptible.

3.3.1. Complémentarité rythmique avec la réponse

L'essence du contrepoint est le flot rythmique ininterrompu et régulier, c'est-à-dire que les motifs rythmiques des voix sont complémentaires et les écouter en même temps donne l'impression que la musique ne s'arrête pas, certains rythmes ont lieu au même moment pour créer un accord et amplifier la polyphonie.

C'est dans ce but que l'on va définir une liste des motifs rythmiques pour la réponse et proposer des rythmes complémentaires pour l'accompagner. Il y aura parfois plusieurs motifs rythmiques complémentaires possibles pour le même motif initial : il faudra alors choisir de façon aléatoire, ce qui fait que deux contrepoints sur le même sujet de fugue pourront être différents. Pour simplifier le code, un motif rythmique est une succession de rythmes au sein d'une même mesure à 4 temps (c'est pourquoi la somme des rythmes pour chaque motif fait toujours 4).

```
2.0,2.0 | 1.0,0.5,0.5,0.5,0.5,1.0 - 1.0,1.0,1.0,1.0
2.0,1.0,1.0 | 1.0,1.0,2.0
1.0,1.0,1.0,1.0 | 2.0,2.0
4.0 | 1.0,1.0,1.0,1.0 - 2.0,2.0
2.0,0.5,1.5 | 1.5,0.5,2.0
2.5,0.5,0.5,0.5 | 0.5,0.5,0.5,0.5,2.0
```

Figure 3: Motifs rythmiques sur une mesure et leur(s) complémentaire(s)

La **Figure 3** indique quelques motifs rythmiques (à gauche de 'l') et leurs complémentaires (à droite de 'l', séparés par des tirets) que l'utilisateur peut modifier (motifs stockés dans un fichier texte). Pour chaque mesure, c'est à dire chaque motif rythmique de 4 temps, on prend au hasard une de ses mesures complémentaires que l'on place à la suite d'une liste de rythmes, le tout formant alors les rythmes de notre contre-sujet (voir **Algorithme 2**).

3.3.2. Enchaînement mélodique des notes dans le contre-sujet et harmonie avec la réponse

On définit l'**intervalle harmonique** comme étant la distance entre deux notes jouées simultanément et un **intervalle mélodique** comme étant la distance entre deux notes jouées l'une après l'autre. Un intervalle harmonique

Algorithme 2 : EcrireRythmeCS(S : sujet, $contreMesure$: tableau de durées de notes en **Figure 3**) : rythmes du contre-sujet

Variables : r : rythmes du contre-sujet
$r \leftarrow []$;
pour chaque *mesure* m *de* S **faire**
  $i \leftarrow aleatoire(0, taille(contreMesure[m]) - 1)$;
  $r \leftarrow ajouterFin(contreMesure[m][i])$;
fin
renvoyer r;

est toujours positif ($i_h = |h_1 - h_2|$) mais un intervalle mélodique est positif si la première note est plus grave que la deuxième et négatif si la première note est plus aiguë que la seconde ($i_m = h_2 - h_1$). *Les intervalles sont mesurés en demi-tons.*

```
vector<int> gammeMajeure = {0,2,4,5,7,9,11};
vector<int> gammeMineure = {0,2,3,5,7,8,11};
vector<int> lIntervalH = {3,4,5,7};
vector<int> lIntervalM = {-2,2,1,-1,-3,5,3,4,-4,-5,0};
vector<int> lIntervalM2 = {-3,-2,4,-4,1,-1,5,3,-5,2,0}; (Marche Harmo)
```

Figure 4: Extrait du code C++ présentant les intervalles harmoniques et mélodiques prioritaires et les intervalles de chaque degré des gammes (majeure ou mineure) avec la tonique (intervalles en demi-tons)

Dans la **Figure 4**, les variables *gammeMajeure* et *gammeMineure* représentent les distances mélodiques de chaque note des gammes à leur tonique respective. On obtient les gammes dans les tonalités voulues en ajoutant la valeur de la hauteur de la tonique en demi-ton par rapport à *Do* (par exemple, $\forall n \in$ *gammeMajeure*, $\forall x \in \mathbb{N}^*$, $n + 5 + 12x$ sont toutes les notes de la gamme de *Fa Majeur* sur toutes les octaves car 5 correspond à *Fa*).

La variable *lIntervalH* correspond à la liste de tous les intervalles harmoniques possibles pour deux notes données (tierce mineure et majeure, quarte et quinte qui sont plutôt consonants). Les listes *lIntervalM* et *lIntervalM2* sont deux ordres de préférence d'intervalles mélodiques pour deux notes données. On privilégie les petits intervalles mélodiques (on ne discute pas ici du signe) car ils sont plus naturels dans une composition, une mélodie étant généralement composée de notes qui se suivent en hauteur, les intervalles plus grands étant utilisés si les petits ne sont pas possibles (approche intuitive). Pour un rythme fixé on obtient donc toujours la même génération de contre-sujet.

L'**Algorithme 3** permet de savoir quelle est la note sur une voix à un instant t (t-ième temps du morceau). Cette fonction sera très utile pour avoir les informations sur une note dont on veut calculer un intervalle harmonique avec son homologue, *i.e*, la note simultanée sur l'autre voix.

Ensuite, l'**Algorithme 4** va choisir des hauteurs pour les notes du contre-sujet (sur les rythmes préalablement calculés) en respectant les contraintes sur la tonalité du morceau, les intervalles harmoniques cohérents avec les notes de la réponse et l'ordre de priorité des intervalles mélodiques (les plus petits d'abord, les plus grands ensuite, l'unisson en dernier) pour garantir une uniformité de la mélodie du contre-sujet et de son harmonie avec la réponse.

Algorithme 3 : NoteSelonTemps($voix$: tableau de notes, t : durée) : note

Variables : i : entier, $sommeT$: durée
$i \leftarrow 0$; `// Variable qui contiendra le numéro de la note sur la voix`
$sommeT \leftarrow 1.0$; `// Variable qui contient la somme des rythmes des notes visitées, début sur le` **`premier temps`** `de la mesure 1`
tant que *sommeT* $<$ *t* **faire**
  $sommeT \leftarrow sommeT + voix[i].rythme$;
  $i \leftarrow i + 1$;
fin
si *sommeT* $\neq$ *t* **alors**
  $i \leftarrow i - 1$;
fin
renvoyer $voix[i]$;

Algorithme 4 : EcrireHauteurCS(CS : sortie de l'**Algorithme 2**, R : sortie de l'**Algorithme 1**) : contre-sujet

Variables : CS : contre-sujet
pour chaque *note* n **de** $rythmeCS$ **faire**
  $m \leftarrow$ note correspondante dans R;
  $nPred \leftarrow$ note précédente dans CS ou dans S si c'est la première note;
  pour chaque im **de** $lIntervalM$ **faire**
    si *compatible(nPred+im,m,lIntervalH)* **alors**
      si $im \neq 0$ **alors**
        $n.hauteur \leftarrow nPred.hauteur + im$
      sinon
        rallonger $nPred$;
      fin
    sinon
      $n \leftarrow$ silence;
    fin
  fin
fin
renvoyer CS;

La variable *lGammeMode* correspond à la liste des intervalles à la tonique dans la gamme et le mode choisi. Les données *lIntervalM* et *lIntervalH* sont respectivement l'ordre de préférence des intervalles mélodiques et harmoniques. Pour chaque intervalle mélodique, on vérifie si la note est bien dans la gamme correspondant à la tonalité du morceau et si elle est bien en harmonie avec son homologue sur l'autre voix. On choisit la première qui est sous ces contraintes. Si la hauteur choisie est la même que

la précédente (unisson mélodique), on fusionne ces deux notes pour en donner une seule dont la longueur est la somme. Si on ne trouve pas de hauteur possible sous ces contraintes, on écrit un silence de longueur égale à la note que l'on aurait dû placer.

3.4. Marche harmonique

Une **marche harmonique** est un motif rythmique et mélodique (parfois commun à plusieurs voix) appelé **modèle**, répété plusieurs fois de suite mais transposé à chaque répétition (ou reproduction) qui peut servir de transition entre des répétitions de sujets et de contre-sujets. Elle peut appartenir à ce que l'on appelle le **divertissement** (voir **Figure 5**). Lorsque le motif est répété mais transposé sur un degré plus aigu, on parle de *marche harmonique ascendante*. Lorsqu'au contraire il est de plus en plus grave, c'est une *marche harmonique descendante* (**Figure 6**). Chaque reproduction du modèle est généralement répétée un ton ou un demi-ton plus haut ou plus bas que la précédente selon le sens de la marche choisi [4]. Le nombre de reproductions sera un entier aléatoire entre 2 et 4 car c'est ce nombre là qu'on trouve généralement dans les fugues de J. S. Bach, ce qui équivaut à des marches harmoniques de longueur 3 à 5 car on a $l_{marche} = n_{reproduction} + 1$).

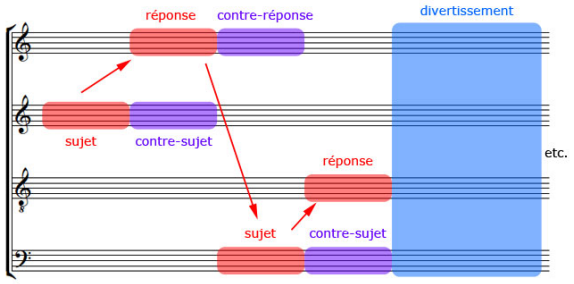


Figure 5: Schéma simplifié d'une fugue

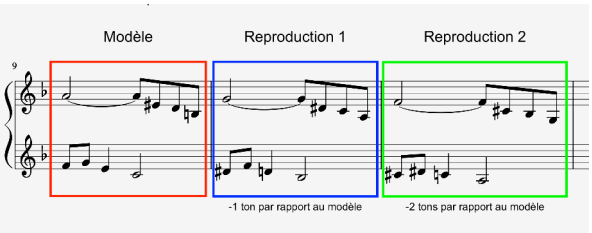


Figure 6: Exemple d'une marche harmonique descendante (par tons) générée automatiquement avec un motif complémentaire sur deux voix (sur la base du premier rythme de la **Figure 7**)

Pour simplifier dans le cas d'une fugue à 2 voix, on considère que le motif fait 4 temps (une mesure pleine).

4 . Dans le cadre de notre prototype, on génère une marche harmonique non-modulante. Si on avait voulu améliorer notre algorithme pour générer des marches modulantes, on aurait fait en sorte de moduler dans la bonne tonalité à chaque reproduction.

```
0.5,0.5,1.,2. | 2.5,0.5,0.5,0.5
0.5,0.5,1.5,0.5,1. | 1.5,0.5,1.5,0.5
0.5,0.5,0.5,0.5,2. | 1.,1.5,0.5,0.5,0.5
```

Figure 7: Motifs rythmiques de marche harmonique sur une mesure et leur complémentaire

Algorithme 5 : EcrireMarcheH($voix1$, $voix2$: tableaux de notes, $lGammeMode$, $lIntervalM2$, $lIntervalH$: liste de hauteurs, $motifMarcheH$: map<listes de durées de notes, listes de listes de durées de notes>) : tableau de tableaux de notes

Variables : $lHautGamme$: liste de hauteurs de notes, $decalage$,r,v,nb : entiers, $interv$,h,i : hauteur, $rythmeV1$,$rythmeV2$: liste de durées de notes, $nPred$,n : note, $motif1$, $motif2$: tableau de notes

$lHautGamme \leftarrow lGammeMode$;
pour chaque *entier* e **de** $lHautGamme$ **faire**
  $lHautGamme \leftarrow lGammeMode + hauteurTonique$; // Attribution de la bonne gamme
fin
$decalage \leftarrow aleatoireEntre(0,1)$; // Sens de la marche harmonique
$interv \leftarrow aleatoireEntre(1,2)$; // Intervalle de reproduction de la marche harmonique
si $decalage = 1$ **alors**
  $decalage \leftarrow interv$; // Marche harmonique montante
sinon
  $decalage \leftarrow -interv$; // Marche harmonique descendante
fin
$r \leftarrow aleatoireEntre(0, motifMarcheH.taille - 1)$; // Choix aléatoire du motif de la marche
$v \leftarrow aleatoireEntre(0,1)$; // Choix aléatoire de la voix pour le motif ou son complémentaire
$rythmeV1 \leftarrow motifMarcheH[r][v]$;
$rythmeV2 \leftarrow motifMarcheH[r][1-v]$;
$nb \leftarrow aleatoireEntre(2,4)$; // Choix aléatoire du nombre de reproductions
pour chaque *flottant* f **de** $rythmeV2$ **faire**
  $nPred \leftarrow voix2[voix2.taille - 1]$; // Note précédant la marche sur la voix 2
  $n \leftarrow nouvelleNote()$;
  $n.rythme \leftarrow f$;
  $h \leftarrow nPred.hauteur$;
  tant que $(h = nPred.hauteur)$ ***ou***
  $(h \bmod 12 \notin lHautGamme)$ **faire**
    $i \leftarrow aleatoireEntre(0, lIntervalM.taille - 1)$; // Intervalle mélodique aléatoire
    $h \leftarrow nPred.hauteur + lIntervalM$; // Calcul de la nouvelle hauteur
  fin
  $n.hauteur \leftarrow h$; // Attribution de la bonne hauteur à la note
  $ajouterFin(voix2, n)$;
  $ajouterFin(motif2, n)$; // Extraction du motif sur la voix 2
  $nPred \leftarrow n$;
fin
$voix1 \leftarrow EcrireHauteurCS(voix1, voix2, lGammeMode, lIntervalM, lIntervalH, 4, tonique)$;
pour i ***de*** $rythmeV1.taille - 1$ ***à*** 0 **faire**
  $ajouterFin(motif1, voix1[voix1.taille - 1 - i])$; // Extraction du motif sur la voix 1
fin
renvoyer
  $EcrireReprod(voix1, voix2, motif1, motif2, decalage, nb)$; // Reproductions

On construit une liste de motifs sur deux voix (**Figure 7**) séparés par un 'l' pour ce qui est des motifs complémentaires d'une même marche et séparés par un retour à ligne pour les différents éléments de cette liste. On choisit aléatoirement le sens de la marche (montante ou des-

cendante), l'intervalle de reproduction (un demi-ton ou un ton), le nombre de reproductions (de 2 à 4), le motif de la marche et l'attribution d'une partie de ce motif à une voix et l'autre partie à la seconde voix. Ensuite, on attribue une mélodie sur ces rythmes pour la voix qui vient d'énoncer la réponse. On choisit cette mélodie en sélectionnant un intervalle mélodique aléatoire qui n'est pas un unisson pour chaque note que l'on place. On écrit ensuite l'autre voix de la même manière qu'on a attribué des hauteurs de note lors de la génération du contre-sujet (en prenant comme réponse la partie de la marche harmonique déjà attribuée). On a alors écrit le modèle de notre marche harmonique (déroulé sur **Algorithme 5**).

3.5. Suite du contrepoint, succession de sujets, contre-sujets, réponses et marches harmoniques

Une **fugue**, vue sous l'angle structurel ici privilégié, est donc une succession d'entrées de sujets, de réponses et de contre-sujets sur un certain nombre de voix. On considère qu'un sujet ne peut être ré-exposé à l'identique, transposé et/ou avec de légères variations de notes et de rythmes que lorsque la réponse de l'exposition précédente suivie d'une marche harmonique est terminée. Sur 2 voix, ce principe parait trivial mais il l'est beaucoup moins lorsqu'il s'agit de faire la même chose pour un plus grand nombre de voix. La **Figure 8** représente les premières séquences de la toccata et fugue BWV 538 [5] .

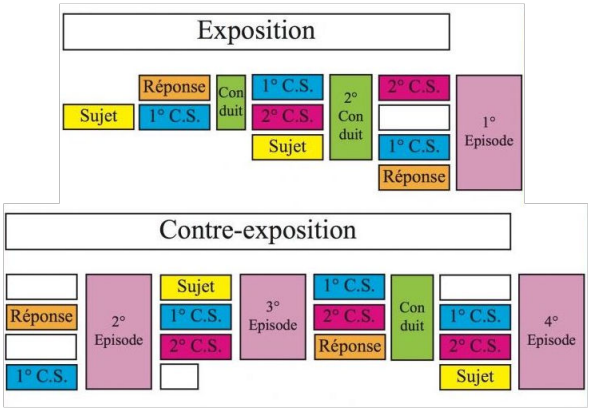


Figure 8: Plan du début de la fugue à 4 voix issue de la Toccata et Fugue en *Ré mineur* (*BWV 538*) de J. S. Bach, extrait de *https://jplecaudey.com/analyse-musicale/bwv-538-toccata-e-fuga-in-d/*

Un **conduit** correspond à une marche harmonique et fait le lien entre deux expositions du sujet voire parfois entre un sujet et contre-sujet ou encore entre deux contre-sujets. Les **épisodes** sont des segments de la composition qui peuvent eux aussi contenir des marches harmoniques mais également des méthodes d'écriture non abordées ici mais qui ne sont pas moins programmables (comme par exemple des passages à la tonalité relative).

Les *motifs* (rythme et hauteur de notes) composant le modèle sur les 2 voix sont stockés dans les variables *motif1* et *motif2*. L'**Algorithme 6** détaille comment le motif est répété *nb* fois (nombre de reproductions) sur les 2 voix en prenant en compte le décalage et en transposant de plus en plus le motif original. Les notes résultant de ces reproductions successives sont ajoutées aux tableaux de notes correspondant aux mélodies des 2 voix. L'algorithme renvoie un tableau contenant les notes des deux voix.

Algorithme 6 : EcrireReproductions($voix1$, $voix2$,$motif1$,$motif2$: tableaux de notes, $decalage$: hauteur, nb : entier) : tableau de tableaux de notes

Variables : $decalageTotal$: hauteur, n : note
$decalageTotal \leftarrow 0$; `// Initialisation du décalage`
pour *i* ***de*** 1 ***à*** nb **faire**
  $decalageTotal \leftarrow decalageTotal + decalage$; `// Incrémentation du décalage`
  pour chaque *note* $n1$ **de** $motif1$ **faire**
    $n \leftarrow nouvelleNote()$;
    $n.rythme \leftarrow n1.rythme$; `// Conservation du même rythme sur la voix 1`
    $n.hauteur \leftarrow n1.hauteur + decalageTotal$; `// Transposition du motif 1`
    $ajouterFin(voix1, n)$;
  fin
  pour chaque *note* $n2$ **de** $motif2$ **faire**
    $n \leftarrow nouvelleNote()$;
    $n.rythme \leftarrow n2.rythme$; `// Conservation du même rythme sur la voix 2`
    $n.hauteur \leftarrow n2.hauteur + decalageTotal$; `// Transposition du motif 2`
    $ajouterFin(voix2, n)$;
  fin
fin
renvoyer $nouveauTableau(voix1; voix2)$;

Dans les fugues plus simples (**Figure 5**), on a bien sûr l'exposition du sujet, la réponse et le contre-sujet (parfois une contre-réponse qui est complémentaire au sujet non transposé). La suite de l'oeuvre est appelée *divertissement* (terme déjà abordé lors de la définition des marches harmoniques) et on n'y retrouve pas de motifs clairement liés au sujet, il s'agit plutôt là de variations comme des marches harmoniques isolées ou d'autres techniques d'écriture que nous ne traiterons pas ici.

3.6. Fin d'un contrepoint

Les longueurs des fugues tout comme le nombre de voix sont variables, allant de quelques dizaines de mesures à parfois des centaines. On donne au programme la contrainte de terminer le morceau après un certain nombre de mesures (le programme peut dépasser ce nombre de

5 . Pour écouter l'œuvre : *https://youtu.be/ZRY7zrMGCi8?t=311*

mesures le temps de terminer le motif qu'il est en train de générer). On veut conclure le morceau sur une **cadence parfaite**, c'est à dire un enchaînement du cinquième degré au premier degré. Lorsque la tonalité d'un morceau est dans le mode mineur, il est de coutume (dans la musique baroque notamment) de terminer le morceau sur la même tonique mais dans le mode majeur. C'est ce que l'on appelle une **tierce picarde**.

3.7. Discussion sur les algorithmes

On remarque que dans la **Figure 9**, la complexité de l'écriture de la fugue complète est de $O(t^{nbVoix})$ (avec t le nombre maximum de notes par voix et $nbVoix$ le nombre de voix). Lorsqu'on travaille avec deux voix (comme tout au long de cette étude), la complexité est de $O(t^2)$. La complexité de notre algorithme d'écriture de contrepoints est donc raisonnable pour un faible nombre de voix mais explose si on en ajoute d'autres. De plus, on ne peut pas facilement adapter notre algorithme pour plus de 2 voix car il ne suffit pas d'appliquer les contraintes sur les voix 2 à 2, il faut le faire sur toutes les voix en même temps (notamment les contraintes liées aux intervalles harmoniques).

Plusieurs de ces complexités dépendent de la longueur des listes comprenant les intervalles mélodiques et harmoniques ainsi que le nombre de notes dans la gamme. Ces termes étant constants, on pourrait les simplifier dans l'expression des complexités pour clarifier l'écriture (c'est le cas des **Algorithmes 4, 5 et 5'**).

	Nom	Structure	Composante note	Complexité
1	EcrireReponse	Réponse	Rythme/Hauteur	$O(n)$
2	EcrireRythmeContreSujet	Contre-sujet	Rythme	$O(m)$
3	NoteSelonTemps		$\emptyset$	$O(t)$
4	EcrireHauteurContreSujet		Hauteur	$O(m * i_{\text{mel}} * i_{\text{har}} * t_{\text{gam}})$
5	EcrireMarcheHarmonique	Marche harmonique	Rythme/Hauteur	$O(f * i_{\text{mel}} * t_{\text{gam}})$
5'	EcrireModele			$O(f * i_{\text{mel}} * t_{\text{gam}})$
6	EcrireReproductions			$O(\text{nb} * f)$
7	EcrireFugue	Fugue entière		$O(t^{\text{nbVoix}})$

Légende

- n est le nombre de notes dans le sujet
- m est le nombre de notes dans le contre-sujet
- t est le nombre de notes sur la voix jusque-là (au maximum, nombre de notes sur toute la voix)
- i_{mel} est le nombre d'intervalles mélodiques
- i_{har} est le nombre d'intervalles harmoniques
- t_{gamme} est le nombre de notes dans la gamme de la tonalité choisie
- f est le nombre maximal de notes sur un motif de marche harmonique
- nb est le nombre de reproductions d'une marche harmonique (au maximum 4)
- $nbVoix$ est le nombre de voix

Figure 9: Complexité des différents algorithmes étudiés (l'algorithme 5' est l'algorithme 5 sans compter l'appel à l'algorithme 6)

4. THÉORISATION GÉNÉRALE D'UN CONTREPOINT PAR UN HMM

Pour théoriser plus facilement et fidèlement les différents éléments d'une fugue, on construit une ébauche de modèle de Markov caché ou HMM (pour *Hidden Markov Model*) (**Figure 10**) élaboré à partir d'une analyse musicale manuelle traditionnelle de quelques fugues du *Clavier Bien Tempéré I*. L'**Algorithme 7** utilise cet HMM et choisit de prendre telle ou telle transition vers un autre état (chaque état est un algorithme comme ceux qui ont été détaillés précédemment). L'état final est atteint lorsque le nombre de mesures escompté pour le morceau est égalé ou légèrement dépassé.

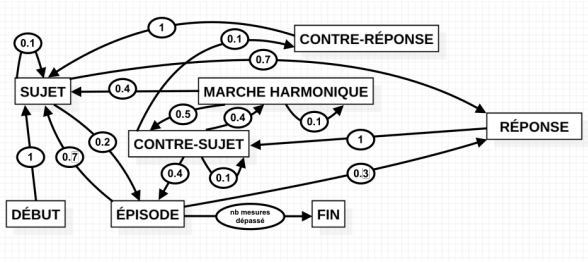


Figure 10: Ébauche de modèle de Markov caché d'une fugue

Algorithme 7 : EcrireFugue($voix1$,$voix2$, $sujet$: tableaux de notes, $nbMesuresMax$: entier, $modele$: HMM) : tableau de tableaux de notes

Variables : $nbMesures$: entier, n : note, $etatHMM$: état en cours dans le HMM, $terminaison$: booléen

$nbMesures \leftarrow 0$; `// Initialisation du nombre de mesures déjà écrites`

$etatHMM \leftarrow DEBUT$; `// Initialisation de l'état HMM`

$terminaison \leftarrow faux$;

tant que *non(terminaison)* **faire**

  $executerEtat(etatHMM, modele)$;

  $nbMesures \leftarrow nbMesures + nbMesuresCreees()$; `// Incrémentation du nombre de mesures`

  $etatHMM \leftarrow choisirEtatSuivant(etatHMM, modele)$;

  si $(etatHMM = EPISODE)$ ***et*** $nbMesures \geq nbMesuresMax$ **alors**

    $etatHMM \leftarrow FIN$;

    $executerEtat(etatHMM)$; `// Composer la fin du morceau`

    $terminaison \leftarrow vrai$;

  fin

fin

renvoyer $nouveauTableau(voix1; voix2)$;

Afin d'affiner les valeurs des transitions du modèle, nous analyserons dans la prochaine section les données publiées par Giraud *et al.* dans les articles déjà cités, et tenterons d'amender le modèle traditionnel avec les résultats de l'analyse statistique du corpus de fugues.

5. ANALYSE ET UTILISATION DES DONNÉES RÉCOLTÉES PAR L'ÉQUIPE ALGOMUS

La plateforme d'annotation et de visualisation *Dezrann* [6] proposé par Giraud *et al.* permet les actions suivantes :

- la reconnaissance des différents éléments d'une fugue
- une étude de la superposition des éléments (entre les voix)
- une étude de la succession des éléments
- une étude fréquentielle des transitions possibles entre éléments.

Nous avons réalisé l'ensemble de ces actions sur les 24 fugues fournies par l'équipe, mais pour des raisons de concentration sur le modèle HMM proposé préalablement, nous nous limiterons à l'étude des transitions pour lesquelles nous avions défini des valeurs à partir d'observations de faible ampleur.

Les données à analyser se présentent sous forme d'un fichier JSON pour chacune des 24 fugues : il comprend la liste des éléments identifiés dans la fugue concernée avec pour chacun d'entre-eux le type, le temps de départ (peut commencer sur un quart de temps ou un demi-temps), leur durée, le numéro de la voix sur laquelle il est (compté de haut en bas) et des informations plus précises sur l'élément en question.

5.1. Analyse de la transition entre les éléments d'une fugue

On cherche à étudier la manière dont les éléments précédemment évoqués (*i.e*, les sujets, réponses *etc.*) se succèdent dans les fugues du *Clavier Bien Tempéré*. Pour cela on écrit différents algorithmes qui ne sont pas détaillés ici [7].

On récolte tout d'abord les statistiques sur les transitions entre les éléments constituants des fugues 2 à 2 ainsi que lorsque l'un de ces éléments est le dernier de la voix ($'FIN'$). On note le nombre total de transitions depuis chaque élément dans la valeur $'TOTAL'$. Cet algorithme récolte donc les données pour les transitions au départ d'un élément e pour toutes les voix de l'ensemble des 24 fugues. Un autre algorithme permet de créer une matrice d'entiers en appliquant le précédent algorithme autant de fois que d'éléments à prendre en considération. Ces entiers représentent le nombre de transitions entre un élément et un autre. On utilise les données précédentes pour calculer les probabilités de transition (tableau de flottants). Pour chaque case, il fait le calcul (1) avec x et y 2 éléments, $x \rightarrow y$ la transition de x vers y et $x \rightarrow$ les transitions depuis x. On obtient ainsi une matrice de transition entre les différents éléments constituant une fugue que l'on va par la suite représenter par un HMM.

$$P(x \rightarrow y) = \frac{nb(x \rightarrow y)}{nb(x \rightarrow)} \quad (1)$$

5.2. Discussion sur ces résultats et comparaison avec le modèle traditionnel

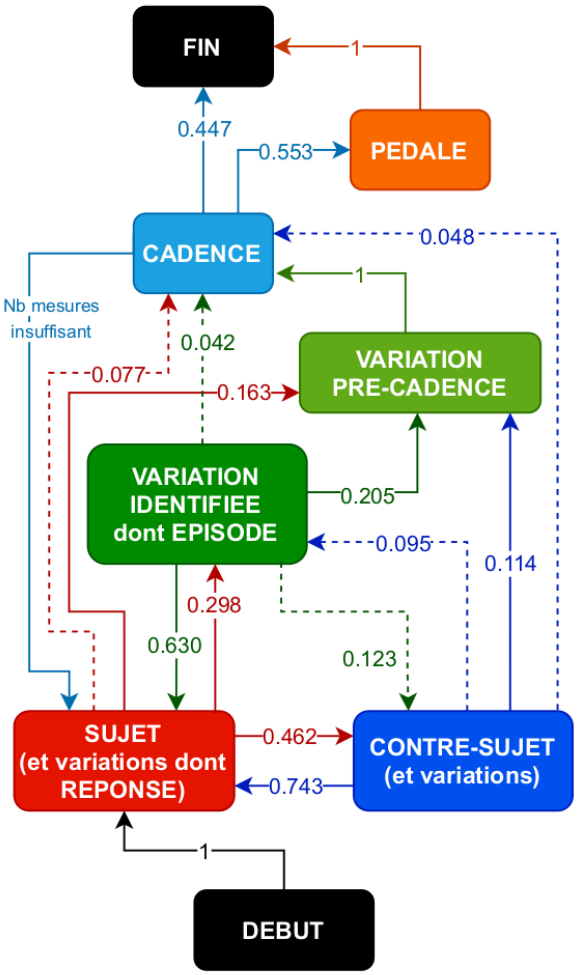


Figure 11: Modèle de Markov caché (HMM) des transitions entre les différents éléments d'une fugue selon l'analyse des fugues du *Clavier Bien Tempéré I* par Giraud *et al.*

L'environnement *Dezrann* modifie complètement le modèle traditionnel général de structuration et d'enchaînement de la fugue pour la génération. Tous les éléments modificatifs ne sont pas ici reportés, mais il est intéressant de voir ce qui, à configuration similaire, a été véritablement mis à jour. Pour cela, nous avons refait une représentation en graphe de notre HMM en calculant les valeurs des transitions à partir des données de Giraud *et al.* (voir **Figure 11**). Nous nous sommes autorisés à retourner depuis une cadence vers l'exposition d'un sujet (ou variation du sujet) à condition que le nombre de mesures minimal (donné par l'utilisateur) n'ait pas été atteint. On

6 . http://algomus.fr/dezrann/, http://www.dezrann.net/

7 . Le détail de ces algorithmes est disponible sur le *Git* : *https://gitlab.etu.umontpellier.fr/jeremieroux/bach* notamment sur la version 4 du rapport

a également modifié le HMM de façon qu'il n'y ait pas de boucle (transition d'un état x vers lui même) : ce problème réside sans doute dans le fait que l'algorithme de reconnaissance indexe les variations des éléments et que lorsque l'on supprime ces indexations (pour la clarté du HMM) on a deux éléments identiques qui se succèdent sur la même voix. Nous avons aussi confondu toutes les variations d'un même élément pour des raisons d'identification mais aussi pour pouvoir jouer plus tard sur l'évolution de l'élément original au cours de la musique en définissant d'autres ordres et règles propres à chaque élément. Les probabilités ne sont donc plus vraiment comparables même si l'on remarque en avoir sous-estimé quelques-unes comme le retour (sur une voix donnée) d'un contre-sujet vers un sujet (et leurs variations) que l'on peut expliquer par la présence dans le corpus de retours au sujet sans développement préalable. Cette analyse a aussi permis d'identifier d'autres éléments comme les pédales et les cadences ainsi que leurs transitions associées.

5.3. Génération de squelettes de fugues selon le HMM de transition des éléments calculé

On utilise maintenant les données présentées en **Figure 11** pour générer des squelettes de fugue sous le même format que les fichiers fournis par Giraud *et al.* L'**Algorithme 8** étage l'entrée des sujets et fait se succéder les divers éléments. On crée une matrice de probabilités (**Figure 12**) qui intègre les données du HMM dans le code. La **Figure 13** est un exemple de structure de fugue ainsi générée qui étage l'entrée des sujets et fait se succéder les éléments.

```
vector<string> elem = {"S", "CS", "Cad", "Ped", "Var", "?", "FIN"};
vector<vector<float> > trans =
  {{0.000, 0.462, 0.077, 0.000, 0.298, 0.163, 0.000},
   {0.743, 0.000, 0.048, 0.000, 0.095, 0.114, 0.000},
   {0.000, 0.000, 0.000, 0.553, 0.000, 0.000, 0.447},
   {0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 1.000},
   {0.630, 0.123, 0.041, 0.000, 0.000, 0.205, 0.000},
   {0.000, 0.000, 1.000, 0.000, 0.000, 0.000, 0.000}};
```

Figure 12: Tableau des probabilités de transitions et du nom des éléments correspondants

```
vide vide vide    S   CS    S    ?  Var    S  Var  Cad  FIN
vide    S  Var    S   CS    S  Var    S    ?  Var  Cad  FIN
   S    ?  Cad  Var    ?  Var    ?  Var    S  Var  Cad  FIN
vide vide    S   CS    S   CS    S   CS    S  Var  Cad  FIN
```

Figure 13: Exemple de génération d'un squelette de fugue à 4 voix

Algorithme 8 : GénérationSquelette($ordre$: ordre d'entrée des voix, $elem$: liste de string, $trans$: tableau 2d de flottants) : tableau 2d de string

Variables : F : tableau 2d de flottants, lg : entier, $nbElemMin$: entier, $suivant$: string
$F \leftarrow creerTableau2d$(taille($ordre$),0); // Création d'un tableau du bon nombre de lignes vides
$lg \leftarrow 0$; // Nombre d'éléments sur chaque voix
$nbElemMin \leftarrow 7$; // Nombre minimal d'éléments sur chaque voix (alternative au nombre de mesures)
/* Etagement de l'entrée des sujets */
tant que *lg<taille(ordre)* **faire**
  pour i ***de*** 1 ***à*** *taille(ordre)-1* **faire**
    si i=lg **alors**
      ajouter(F[$ordre[i]$],'S');
    sinon
      si i<lg **alors**
        ajouter(F[$ordre[i]$],'vide');
      sinon
        ajouter(F[$ordre[i]$],elemTrans(F[$ordre[i]$][-1],elem,trans));
      fin
    fin
  fin
  $lg \leftarrow lg + 1$;
fin
/* Suite après l'étagement */
tant que *F[0][-1]* $\neq$ *'FIN'* **faire**
  pour i ***de*** 1 ***à*** *taille(ordre)-1* **faire**
    /* On calcule l'élément suivant */
    $suivant \leftarrow$elemTrans(F[$ordre[i]$][-1],elem,trans);
    si $suivant$='Ped' || $suivant$='FIN' **alors**
      si lg<$nbElemMin$ **alors**
        ajouter(F[$ordre[i]$],'S');
      sinon
        /* Fin de la fugue */
        ajouterToutesLignes(F,$suivant$);
        si $suivant$='Ped' **alors**
          ajouterToutesLignes(F,'FIN');
        fin
        $i \leftarrow$taille($ordre$)-1;
      fin
    sinon
      ajouter(F[$ordre[i]$],$suivant$)
    fin
  fin
  $lg \leftarrow lg + 1$;
fin
renvoyer F;

6. CONCLUSION

La fugue est le type de morceau qui représente le mieux l'écriture du contrepoint chez Jean-Sébastien Bach. Polyphonique par essence, elle présente une structure qui se prête bien à une démarche algorithmique de construction, aussi bien qu'à une analyse automatique des motifs qui la constituent.

Notre travail, ici présent, a consisté à proposer un ensemble d'algorithmes d'écriture des éléments d'une fugue, à assembler pour réaliser un générateur automatique de fugues à la *J. S. Bach*. Après avoir brièvement passé en revue les travaux les plus proches de notre objectif, nous nous sommes centrés sur les éléments constitutifs des fugues de Bach. La section 3 a décrit ces différents constituants : le sujet, ensemble mélodique de base servant de motif initial, ainsi que les réponses (motifs complémentaires issus d'une action de transposition harmonique ou rythmique).

Nous avons montré comment on pouvait les représenter en machine, les coder avec un ensemble de contraintes qui ont été énoncées, et les installer sur des fugues à deux voix (limite due à des raisons de faisabilité pour l'état actuellement prototypal du générateur). Sujet et réponse sont aussi suivis de contre-sujets, contre-réponses, également générés. Nous avons aussi montré comment il était possible de produire ces "entre deux" que sont les divertissements, ou épisodes, au sein desquels nous avons cependant distingué un type particulier, la marche harmonique, que nous avons modélisée et ajoutée à notre bibliothèque d'algorithmes. L'enchaînement des différents constituants obéit aussi à un certain nombre de règles à modéliser en génération. La section 4 propose un modèle de Markov caché sous forme d'un graphe dont les noeuds sont les constituants et les arcs représentent des transitions ainsi que leurs probabilité de franchissement, réalisé à partir d'une analyse musicale manuelle traditionnelle. Afin d'affiner ce modèle, nous avons exploité les données de Giraud *et al.* de l'équipe *Algomus*, qui a produit un environnement fort intéressant pour l'analyse et la recherche de données des 24 fugues du livre I du *Clavier Bien Tempéré* de J.S Bach.

Ce qui était important pour nous était de voir comment cet environnement pouvait amender notre modèle d'enchaînement pour la génération. Plusieurs éléments peuvent être exploités, mais nous nous sommes ici surtout focalisés sur leur apport pour une meilleure définition des probabilités de transition. La section 5 montre justement comment ces données sont utilisées ainsi que la modification des valeurs des probabilités de transition entre les constituants. Nous nous sommes également aperçus que l'environnement ainsi exploité nous permettait réellement d'affiner la structure même du graphe d'enchaînement, et pas seulement les probabilités de transition. C'est pourquoi, l'étape prochaine du travail consistera à mettre à jour réellement l'ensemble du graphe d'enchaînement afin de le rendre plus riche et plus conforme aux réelles complexités des fugues de Bach. Ce travail ne peut être possible qu'à l'aide de l'environnement *Dezrann*, [7], car un passage par une analyse manuelle rendrait la tâche beaucoup plus coûteuse en efforts de modélisation et de transposition.

Toujours est-il que ce projet semble converger vers les bases d'un *générateur de fugues*, peut-être restreint dans un premier temps, mais néanmoins faisable. Ce dernier est fondé sur l'exploitation d'une bibliothèque algorithmique de génération de constituants de la fugue par un modèle de type probabiliste pour l'enchaînement desdits constituants. Rien n'empêchera par la suite, une fois le prototype construit, de prévoir des séquences possibles d'apprentissage et d'entraînement sur les fugues analysées par l'équipe *Algomus*, afin de conforter ses choix et/ou de les adapter pour une plus grande conformité avec *L'Art de la Fugue*.

7. REFERENCES

[1] Gédalge André. *Traité de la Fugue*. Enoch Cie, Paris, 1901.

[2] Jean-Sébastien Bach. L'Art de la Fugue (Die Kunst Der Fuge), 1740-1750. Bibliothèque d'État de Berlin.

[3] Gérard Bougeret. Harmonie, contrepoint, représentation : prélude à une didactique de l'écriture musicale. *Spirale. Revue de recherches en éducation, hors-série n4. Les représentations en formation(2), pp. 51-68*, 2005.

[4] Suzanne Clercx. *Le Baroque et la Musique : Essai d'Esthétique Musicale*. Bruxelles, 1948.

[5] David Cope. *Computers and Musical Style*. A-R Editions Inc, Maddison Wisconsin, 1991.

[6] Johann Joseph Fux. *Gradus ad Parnassus*. Johann Peter Van Ghelen, Wien, 1725.

[7] Mathieu Giraud, Richard Groult, and Emmanuel Leguy. Dezrann, a Web Framework to Share Music Analysis. *TENOR, pp. 104-110*, 2018.

[8] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. Computational Fugue Analysis. *Computer Music Journal, 39(2)*, 2015.

[9] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. DeepBach : a Steerable Model for Bach Chorales Generation. *Proceedings of the 34th International Conference on Machine Learning, PMLR 70 :1362-1371*, 2017.

[10] Czaczkes Ludwig. *Analyse des Wohltemperierten Klaviers : Form und Aufbau der Fuge bei Bach*. Österreichischer Bundesverlag, Wien, München, 1965.

[11] Marcel Mesnage and André Riotte. Modélisation informatique de partitions, analyse et composition assistée. In *Cahiers de l'Ircam n 3, Recherche musicale : La composition assistée par ordinateur*, pages 1–1. Ircam - Centre Georges Pompidou, 1993. cote interne IRCAM : Mesnage93a.

[12] Barbaud Pierre. *Initiation à la composition musicale automatique*. Dunod Paris, 1965.

ESPACES SONORES PARADOXAUX : APPROCHE DES HARMONIQUES HYPERSPHÉRIQUES ET IMPLÉMENTATION LOGICIELLE, POUR UNE PRATIQUE CRÉATIVE DE LA SPATIALISATION IMMERSIVE DU SON EN FREE-PARTY

Picciola Daniel
CICM/Musidanse
Université Paris 8
MSH Paris Nord
fenris42@gmail.com

PLAN

Résumé
Introduction

1. À l'écoute des espaces sonores paradoxaux
1.1 À l'écoute des espaces sonores paradoxaux dans la vie quotidienne.
1.2 À l'écoute des espaces sonores paradoxaux en teknival.
1.3 Les espaces sonores paradoxaux: un héritage de Jean-Claude Risset.

2. Approches de l'hyperespace du son
2.1 Les espaces sonores paradoxaux en ambisonie de premier ordre.
2.2 Caractéristiques de l'ambisonie d'ordre supérieur.
2.3 Les espaces sonores paradoxaux dans le domaine sphérique.

3. Le modèle des harmoniques hypersphériques
3.1 Rôle et fonctionnement de l'hyperangle comme paramètre de spatialisation du son.
3.2 Choix du polynôme dans l'encodage des harmoniques hypersphériques.

4. Réactions de la communauté d'écoute

Conclusion
Références

RÉSUMÉ

Cet article commence par définir le phénomène d'espace sonore paradoxal à travers divers types d'écoutes, à la fois dans la vie quotidienne et notamment dans le contexte particulier des *free-parties*, pour expliquer le rôle que je fais jouer à ce phénomène dans ma pratique du *live* en ambisonie.

En fournissant plusieurs exemples de programmations, sous forme de *patches* et d'illustrations sonores, cet article explique d'abord en quoi la composition d'espaces sonores paradoxaux est un héritage des sons paradoxaux de Jean-Claude Risset.

Sont évoquées ensuite les principales étapes de l'élaboration d'espaces sonores paradoxaux en ambisonie d'ordre supérieur.

Cet article présente alors l'étape finale de ma programmation, et mon choix de réaliser les espaces sonores paradoxaux à partir du modèle mathématique des harmoniques hypersphériques. Dans cette étape finale sont détaillées les caractéristiques du résultat sonore que j'obtiens par l'implémentation de ce modèle, et son rôle dans l'utilisation du phénomène d'espace sonore paradoxal comme matériau musical dans mon *live*.

INTRODUCTION

Cela fait maintenant une trentaine d'années que les *free-parties*, originellement appelée *raves*, existent. Elles sont l'occasion d'expérimentations sonores dans la musique électronique populaire. J'y participe depuis la fin des années 90 et je fais partie des rares *livers* qui y jouent en ambisonie.
L'utilisation de nouveaux outils développés en spatialisation du son me permet d'enrichir mon *live* en explorant les possibilités de spatialisation du langage musical tekno des free-parties. Je m'intéresse particulièrement aux traitements ambisoniques d'ordre supérieur, qui font appel au principe des harmoniques sphériques pour décomposer un espace sonore complexe

en une somme d'espaces sonores simples. La forme cardïoïde des harmoniques correspond aux champs de captation d'une somme de microphones virtuels.[1]

En cherchant à appliquer à l'espace sonore le principe des sons et rythmes paradoxaux de Jean-Claude Risset, j'ai imaginé un type de spatialisation où le son semblerait à la fois se rapprocher et s'éloigner, formant un flux continu entre deux sources sonores diamétralement opposées. Pour obtenir ce résultat sonore, j'ai implémenté dans mon *live* le modèle mathématique des harmoniques hypersphériques dont j'aborde le fonctionnement dans cet article. Un nouveau paramètre issu de ce modèle me permet d'opérer des fondus-enchaînés entre deux sources diamétralement opposées: l'hyperangle. Ce nouveau paramètre creuse un silence entre les deux sources en les traitant par annulation de phase. C'est ce traitement par annulation de phase dans le domaine hypersphérique qui caractérise l'effet de fondu-enchaîné entre les deux sources, et par lequel l'espace sonore devient paradoxal.

1. À L'ÉCOUTE DES ESPACES SONORES PARADOXAUX

1.1 À l'écoute des espaces sonores paradoxaux dans la vie quotidienne.

Je décris l'espace sonore paradoxal comme un phénomène psycho-acoustique où un son réfléchi est perçu comme plus proche que sa source, à tel point que leur localisation semble intervertie. Ce phénomène est tellement répandu dans la vie quotidienne que nous n'y prêtons généralement pas attention.
Si vous vivez près d'une voie à grande circulation, dans une ville où les bâtiments sont hauts et larges, offrant ainsi de larges surfaces de réflexion aux bruits environnants, la voie à grande circulation peut vous sembler beaucoup plus proche, et même à l'opposé de sa situation réelle. Si vous circulez en voiture sur cette voie et que vous ouvrez la fenêtre sur votre droite alors qu'un véhicule vous double sur la gauche, votre oreille perçoit qu'il vous double sur la droite. Si vous prenez les transports en commun, que vous attendez sur un quai dans un souterrain à la voûte elliptique, vous pouvez entendre depuis le foyer elliptique où vous êtes une conversation qui a lieu sur l'autre quai, dans le foyer elliptique opposé, comme si vous étiez à coté. On peut observer ce phénomène hors du milieu urbain: je pense aux échos dans la montagne ou au rôle acoustique joué par les falaises sur les bords de mer. En me promenant sur la côte normande à Etretat, je me suis arrêté de profil par rapport à la mer, à mi-chemin entre les deux falaises, dont celle qu'on surnomme l'Aiguille creuse. Je suis resté ainsi pour entendre dans quel sens les vagues s'échouent, et comment arriver à dissocier les réflexions du son des vagues sur les falaises. Ce son, comme le son d'une voie à grande circulation, est proche du bruit blanc, où les fréquences sont indissociables. Le son des vagues émerge d'un champ sonore diffus relativement constant qui s'étend entre les deux falaises ; seules les vagues assez puissantes sont audibles par réflexion, et donc localisables en deux endroits différents, alors que leurs ressemblances sonores tendent à les confondre.

1.2 À l'écoute des espaces sonores paradoxaux en teknival [1].

Le phénomène d'espace sonore paradoxal est courant en teknival, rassemblement de plusieurs *sound systems* en situation de free-party pendant plusieurs jours: la disposition anarchique des murs d'enceintes et des véhicules y crée ce type de phénomène psycho-acoustique. Les sons diffusés et réfléchis sont principalement des pulsations régulières, de vitesses différentes, qui se synchronisent et se désynchronisent périodiquement. Les discrétisations du temps musical par les boîtes à rythmes se superposent en figures rythmiques plus complexes, où il est difficile de dire si les effets de déphasage du type *flanger* proviennent directement de la source ou de son contexte spatial. D'autres sons plus rares, singuliers, plus étranges les uns que les autres, émergent çà et là, entretenant une modulation sonore permanente qui se concentre et se disperse aléatoirement pendant tout le teknival[2]. Les conditions météorologiques interagissent également avec les perturbations atmosphériques sonores émanant du teknival. Quand le vent se lève, chargé de sons, il s'engouffre dans le labyrinthe de murs d'enceintes et de véhicules, et porte parfois le son sur de longues distances, un peu comme un chef d'orchestre qui spatialiserait tout le teknival. Cette interaction entre perturbations atmosphériques de type musical et de type météorologique est propre à l'acoustique des teknivals. Le phénomène d'espace paradoxal, anodin à l'écoute dans la vie quotidienne, peut devenir une condition d'écoute extraordinaire émergeante dans l'espace-temps du teknival.

[1] Roads, Curtis, « l'audionumérique. Musique et informatique », Dunod, 2007 pour la 2e édition, collection audio-photo-video, Chapitre 7 « la spatialisation », p.150-155 « 7.4.3 Ambisonie et synthèse par champ d'onde »

[2] Le titre « Gwerly Mernans », d'Aphex Twin, correspond à cette description du paysage sonore.

Le style de mon *live* poursuit une forme de tekno *acid-core* très expérimentale qui privilégie la sensation de vertige, tout comme les lives de 69db, Curley, Kaos, Explore-toi, Mental perturbations, Les Boucles Étranges, Mem Pamal, LSDF, pour ne citer qu'eux[3].
C'est donc en cherchant des procédés toujours plus chronotropes[4] [2] que j'ai peu à peu intégré à mon *live* les objets de composition développés au CICM[5], notamment concernant le traitement spatial du son.
Cet apport du milieu de la recherche oriente mon *live* vers une esthétique de plus en plus singulière, que j'appelle *science-core*: je programme mon *live* sur MaxMsp[6], où je synthétise tous mes sons par modulation de fréquence[7].

Le CICM développe entre autres des objets de traitement ambisonique de l'espace sonore, que j'implémente dans mon live depuis 2013, en cherchant à composer un type de spatialisation intégré aux codes musicaux reconnus par la communauté d'écoute de free-party, qui est pourtant habituée à la monophonie. Rares sont les *sound-systems* qui pratiquent le *live* en ambisonie: l'un des plus connu est Spiral Tribe, constitué de pionniers anglais du mouvement *free-party*[8]; des *livers* comme 69db, Crystal distorsion et Ixy, ont développé une pratique du *live* en ambisonie. Cette pratique est jusqu'ici trop rare ou mal maîtrisée pour je puisse en avoir déjà fait l'expérience comme auditeur, et il ne semble exister aucun enregistrement: je me contente pour le moment de recueillir des témoignages qui indiquent toutefois que les expériences de diffusion du son en ambisonie remontent jusqu'aux origines des *raves-parties*. L'implémentation des objets de traitement ambisonique dans mon *live* me permet de modéliser le phénomène d'espace sonore paradoxal, afin de me servir de ce phénomène acoustique propre à l'univers de la *free-party* comme matériau musical.

1.3 Les espaces sonores paradoxaux: un héritage de Jean-Claude Risset.

J'ai au départ de mes recherches imaginé le paradoxe sonore appliqué à l'espace dans la continuité des sons et rythmes paradoxaux réalisés par J-C Risset[9]: il faut deux sons identiques pour réaliser un son paradoxal, deux rythmes identiques pour réaliser un rythme paradoxal; il me faut donc deux sources identiques pour réaliser un espace sonore paradoxal. Dans ces trois échelles, les deux matériaux sonores jumeaux suivent une variation d'un état A vers un état B dans la hauteur, le rythme ou l'espace. Cette variation est amplifiée par une enveloppe sinusoïdale dont le début correspond à l'état A et la semi-période à l'état B. Quand l'amplitude d'un des matériaux sonores est pleinement audible, l'autre est silencieuse: les deux éléments sont déphasés selon une symétrie temporelle par laquelle ils se relaient, créant ainsi l'illusion d'une variation infinie sur un même matériau.

C'est durant ma propre expérience sur une spatialisation stéréophonique des sons paradoxaux (fig.1) que l'annulation de phase m'est apparue comme le traitement le plus opérant pour isoler les différences spatiales des composantes sonores et rendre inaudibles les composantes spatiales communes aux deux sources. Les deux hauts-parleurs correspondent à l'état A et l'état B du paradoxe spatial et du paradoxe sonore: avec une forme d'onde en dent de scie, deux fréquences montent ou descendent d'une octave durant toute la traversée de la panoramique. Quand les deux fréquences sont simultanément audibles, à une octave de différence, l'une sur le haut-parleur droit, l'autre sur le haut-parleur gauche, l'annulation de phase creuse un silence dans le centre de l'espace. Le son semble alors plus lointain dans sa provenance et sa destination: le même son semble simultanément s'approcher d'un coté et s'éloigner de l'autre. En traitant deux formes d'ondes sinusoïdales, l'espace sonore peut devenir paradoxal sans recourir à l'annulation de phase. Avec deux formes d'ondes plus complexes, les composantes spectrales communes entrent en annulation de phase. Si les sons traités ne sont pas paradoxaux, le paradoxe spatial est toujours audible grâce à l'annulation de phase.
J'ai alors choisi de reproduire le procédé en ambisonie 2D. C'est ici que l'ambisonie d'ordre supérieur met particulièrement en valeur l'annulation de phase. Le traitement ambisonique diffuse le son par un certain nombre de canaux assignés chacun à une dimension

[3] Beaucoup de leurs lives peuvent être écoutés sur la chaîne « History of Free-Party » https://www.youtube.com/channel/UCYK18wMmP-2EQSZVChp6Yxw

[4] Grisey, Gérard, « Tempus ex-machina, réflexions d'un compositeur sur le temps musical », Entretemps n°8, Paris, 1989.

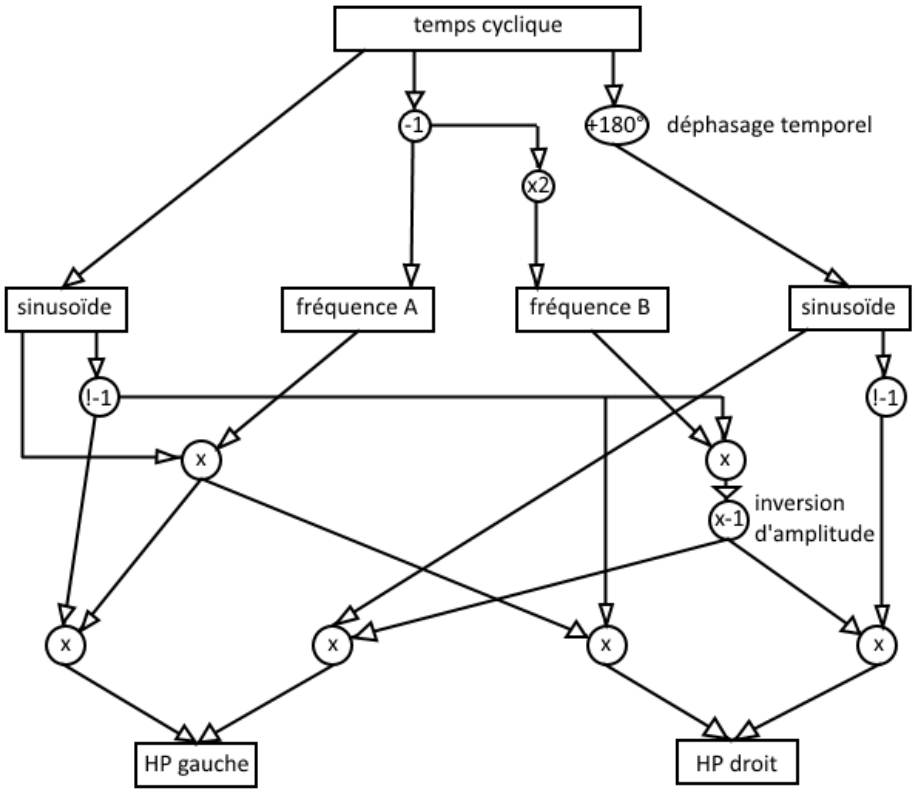
[5] CICM, EA MUSIDANSE de l'université de Paris 8

[6] https://cycling74.com/

[7] https://soundcloud.com/user-956656661/livemshbinaural

[8] https://sp23.org/

[9] Risset, Jean-Claude, « Électron-positron » (commande du CERN pour l'inauguration du LEP), 1989, http://brahms.ircam.fr/works/work/21307/



(fig.1) principe du son paradoxal en stéréophonie

spatiale [3]. Les canaux dont l'ordre est associé à la profondeur sont nommés harmoniques négatifs, les canaux où l'ordre est associé à la largeur sont nommés harmoniques positifs. Les harmoniques négatifs et positifs se regroupent par paires dans des degrés d'ambisonie, comptés de 0 à *n*.

Dans la situation d'annulation de phase entre deux sources sonores diamétralement opposées, chaque canal est alimenté par deux signaux constants, de valeurs absolues égales, l'un est négatif , l'autre positif. La somme des signaux est alors nulle dans les degrés pairs, et seules les différences spatiales du son sont transmises dans les degrés impairs, et diffusées par les haut-parleurs correspondant à l'orientation spatiale de ces canaux ambisoniques. À ce stade, le volume sonore total s'amoindrit, tout en suivant une directionnalité du champ sonore, qui est alors perçu comme étiré dans l'espace.
Dans les degrés pairs l'espace se décompose en un nombre pair d'harmoniques: l'image spatiale est symétrique par rapport au *sweet-spot*. Dans les degrés impairs, l'espace sonore se décompose en un nombre impair d'harmoniques : l'image spatiale est asymétrique par rapport au *sweet-spot*.
Alimentés par les deux signaux constants, opposés et mixés, les harmoniques des degrés impairs contiennent à la fois des valeurs positives et négatives, alors que les harmoniques des degrés pairs ne contiennent que des valeurs soit positives, soit négatives. Ce résultat vient de l'orientation spatiale de chaque harmonique: dans les degrés pairs, l'espace se décompose en un nombre pair de cardioïdes*:* les vecteurs sont traités par des valeurs angulaires entières qui les orientent toujours dans des paires de directions diamétralement opposées. Dans les degrés impairs, l'espace sonore se décompose en un nombre impair de cardioïdes*:* les vecteurs sont traités par des valeurs angulaires semi-entières qui les orientent toujours selon un nombre impair de directions qui sont donc déphasées entre elles, mais jamais opposées.
Dans le sens azimutal, la valeur absolue de l'ordre est égale au degré: l'annulation ou la superposition de

signaux dans les ordres azimutaux s'applique à tout le degré.
Une variable opère le fondu-enchaîné entre les deux sources jusqu'au stade d'espace sonore paradoxal où le son semble s'étirer. Cette variable doit être continue et suivre un mouvement sinusoïdal qui épouse la trajectoire courbée des sources sonores. Dans l'exemple de sons paradoxaux spatialisés en 2D, cette variable est la même pour tous les canaux. Elle prendra par la suite différentes échelles de valeurs selon l'ordre ambisonique, et le fondu-enchaîné pourra s'opérer avec plus de définition spatiale. Je reproduis le procédé pour une diffusion tridimensionnelle (fig.2).

2. APPROCHES DE L'HYPERESPACE DU SON

2.1 Les espaces sonores paradoxaux en ambisonie de premier ordre.

Dans mes premières maquettes logicielles dédiées aux espaces sonores paradoxaux, la paire de sources sonores était traitée par une paire d'objets alors développés au CICM[10]. Ces objets sont *ambipan~* pour une diffusion 2D en quadriphonie, ou *ambicube~* pour une diffusion tridimensionelle en octophonie. L'utilisation de ces objets par paires consiste à répéter deux fois les opérations d'encodage et de décodage, mais en étant limité d'une part au premier degré ambisonique, d'autre part au domaine des ondes planes. L'opération mathématique du décodage en ondes planes consiste à discrétiser une sphère virtuelle en un polyèdre dont chaque angle correspond à un haut-parleur. Reproduire cette opération consiste à reproduire chaque angle du polyèdre, multipliant par deux le nombre de haut-parleurs virtuels.

En cherchant des traitements dans ce double domaine d'ondes planes, j'ai imaginé des cas où les interactions sonores entre deux groupes de haut-parleurs disposés selon les angles de deux polyèdres symétriques pourraient dessiner des figures géométriques plus complexes, décrites dans le film *Dimensions* réalisé par l'ENS de Lyon[11]: ces figures sont formalisables en dimension 4 (figures de dimension 3 multipliées par elles-mêmes), et j'ai cherché dans un premier temps à m'en servir comme modèles pour réaliser des espaces sonores paradoxaux. J'ai donc, avant d'étudier l'encodage, imaginé d'abord un décodage qui discrétiserait une hypersphère en un hyperpolyèdre dont chaque angle serait un haut-parleur virtuel.

2.2 Caractéristiques de l'ambisonie d'ordre supérieur.

L'opération d'encodage consiste à simuler le champ de captation de microphones virtuels orientés dans l'espace selon le modèle mathématique des harmoniques sphériques [4]. Ces harmoniques sont indexés selon un degré *l*, compté à partir de zéro. L'harmonique du degré zéro est un canal monophonique qui désigne l'espace sonore central. À partir du premier degré les harmoniques sont directionnels, et sont indexés selon un ordre *m* qui se calcule de *-l* à *l*. Cet ordre *m* correspond à la dimension spatiale du son. Dans les harmoniques supérieurs, le volume sonore est modéré, et la position du son dans l'espace est différemment déphasée selon l'ordre *m* pour chaque canal: cette superposition de sources modérées et déphasées peut créer un champ sonore complexe qui rend ainsi la spatialisation plus réaliste, notamment par l'annulation de phase entre signaux audio dans les harmoniques pairs quand l'espace sonore est paradoxal. Les traitements ambisoniques d'ordre supérieur de la bibliothèque HOA permettent d'interpoler l'espace sonore composé par sources ponctuelles et l'espace sonore composé par champs sonores.

2.3 Les espaces sonores paradoxaux dans le domaine sphérique.

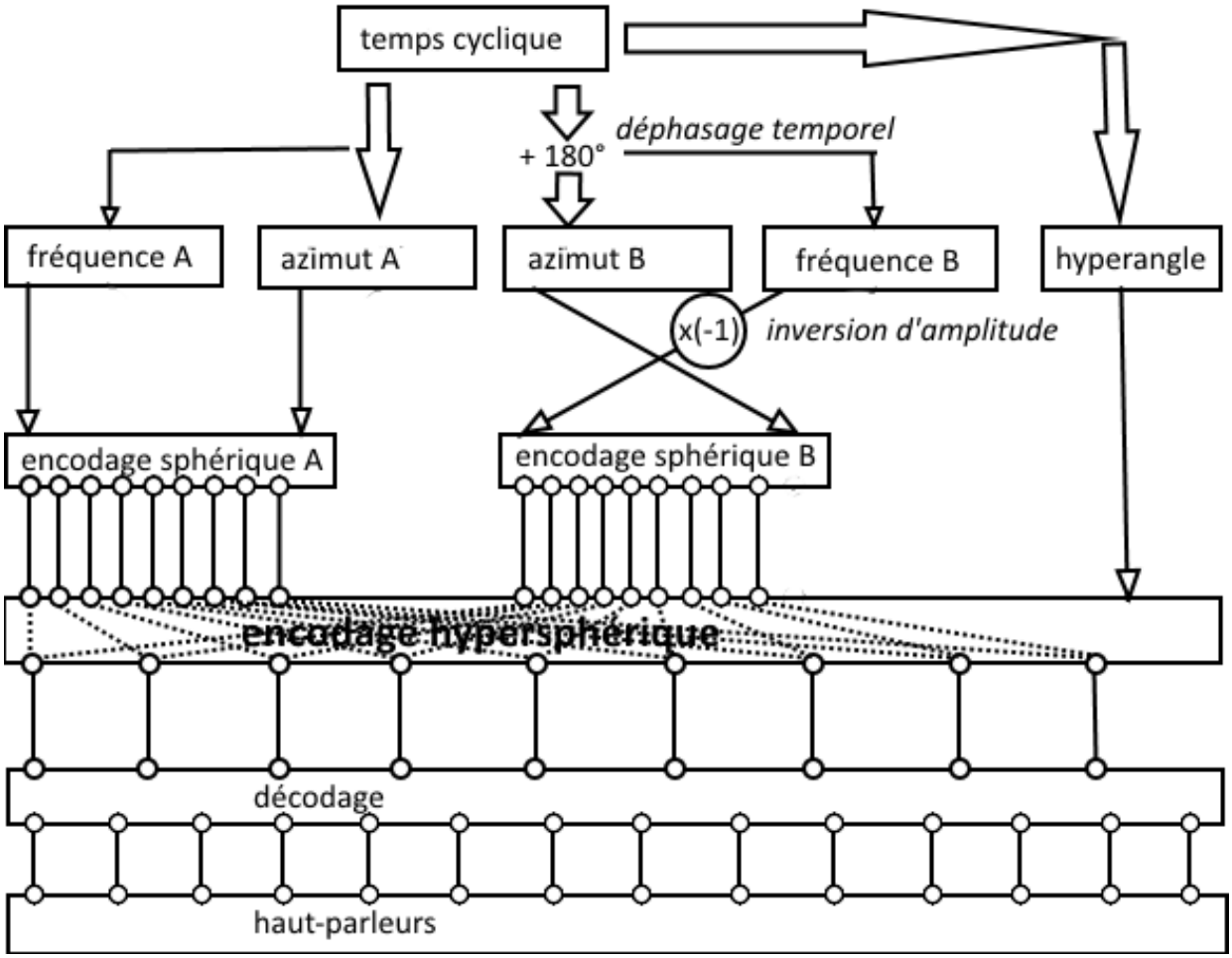
La bibliothèque HOA v1.3.1 permet de traiter l'espace sonore en 2D: les harmoniques sphériques sont tronqués en harmoniques circulaires, qui sont spatialement modulables dans le sens de l'azimut. Les travaux de l'équipe du CICM[12] dans le cadre du projet HOA, ont abouti à la bibliothèque HOA v2.2.[13] Cette bibliothèque a permis des traitements 3D de l'espace sonore: les harmoniques sphériques sont à la fois modulables dans le sens de l'azimut et dans le sens de l'élévation. Des harmoniques intermédiaires interviennent entre chacune

[10] Courribet, Benoît; Paris Eliott; « ambipan~», « ambicube~ », CICM-TOOLS, Université de Paris 8, 2003-2015.

[11] Leys, Jos; Ghys, Etienne; Alvarez, Aurélien; « Dimensions…une promenade mathématique », Creative Commons, 2008. http://www.dimensions-math.org/

[12] . A. Sèdes, P. Guillot, E. Paris, « The HOA Library, rewiew and prospect », actes de l' International Computer Music Conference | Sound and Music Computing 2014, Athènes. Proceedings ICMC| SMC, pp.855 860, 2014, Proceedings ICMC|SMC. <hal-01196453>

[13] J .Colafrancesco, P. Guillot, E. Paris, A. Sedes, A. Bonardi, « La bibliothèque HOA, bilan et perspectives » , JIM 2013, Saint-Denis, actes des Journées d'Informatique Musicale 2013 (JIM 2013), Université Paris 8, Saint-Denis, 2013, p. 187-198 http://www.mshparisnord.fr/JIM2013/actes/jim2013_26.pdf <hal-01196422>



(fig.2) principe du son paradoxal en ambisonie

des trois dimensions, et discrétisent l'espace en autant d'angles que le degré ambisonique l'indique. Le nombre de canaux augmente donc plus avec l'ordre ambisonique 3D qu'il n'augmente en 2D. De nouvelles opérations mathématiques interviennent dans la construction des harmoniques sphériques. La normalisation 3D modère l'amplitude de chaque harmonique par des valeurs constantes calculées selon le degré *l* et l'ordre *m* de chaque harmonique. Le polynôme de Legendre traite chaque harmonique par différentes variables calculées selon les variations de l'élévation: plus le son s'élève, moins il est audible dans le sens de l'azimut, et vice et versa [5].

L'étude du modèle des harmoniques sphériques a orienté mon approche de l'espace en dimension 4 vers un autre modèle, celui des harmoniques hypersphériques, afin de réaliser des espaces sonores paradoxaux en ambisonie d'ordre supérieur par une nouvelle opération d'encodage. Il est pourtant parfaitement possible de les réaliser en utilisant deux fois une opération d'encodage pour opérer un fondu-enchaîné entre deux domaines sphériques, par une variable dont l'échelle change avec l'ordre ambisonique. Se pose alors la question du choix de ces échelles pour cette variable. Cette variable a une importance égale aux variables de l'azimut et du zénith dans le traitement de l'espace sonore que j'utilise pour réaliser des espaces sonores paradoxaux. Envisager cette variable comme une nouvelle dimension possible de l'espace sonore m'amène à dépasser les modèles d'encodage 2D et 3D basés sur les harmoniques sphériques.

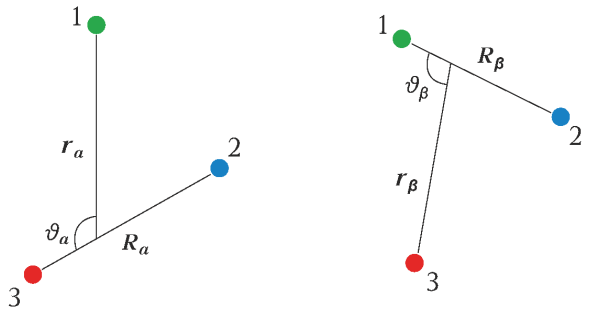
3. LE MODÈLE DES HARMONIQUES HYPERSPHÉRIQUES

3.1 Rôle et fonctionnement de l'hyperangle comme paramètre de spatialisation du son.

Utiliser deux fois le modèle des harmoniques sphériques me permet de traiter l'espace sonore dans un domaine où

chaque harmonique est dédoublé, dans le but d'y opérer le traitement par annulation de phase qui permet d'entendre, comme en Teknival, le même son selon deux sources différentes : j'opte alors pour le modèle des harmoniques hypersphériques (ou HSH, pour *Hyper-Spherical Harmonics*) en programmant un seul objet d'encodage, en 4D ou en 6D. Notons qu'il s'agit d'un modèle utilisé en cosmologie dans la recherche d'atomes exotiques (qui n'existent pas sur Terre, à part dans les collisionneurs de particules). Je reprends ici un article du chercheur A.V Meremianin pour expliquer la construction des HSH [6].

Le modèle mathématique des HSH constitue une approche des systèmes à trois corps à l'échelle quantique: les trois corps étudiés en cosmologie sont les quarks composant les baryons du noyau atomique [14]. L'interaction des trois corps dans l'espace tridimensionnel est alors exprimée par l'interaction entre deux corps et un centre de masse dans un espace en six dimensions. Cet hyperespace est construit par l'addition des deux vecteurs 3D correspondant aux deux corps : en mathématiques, les vecteurs de Jacobi sont deux vecteurs tridimensionnels ayant en commun un centre de masse (équivalent d'un centre de gravité à l'échelle quantique, où d'autres forces fondamentales que la gravité sont à l'œuvre). Les coordonnées des deux vecteurs et du centre de masse se calculent selon une variable nommée hyperangle. L'hyperangle permet de passer des coordonnées du premier vecteur aux coordonnées du second vecteur, par l'intermédiaire des coordonnées du centre de masse (fig.3).



(fig.3) Deux exemples de vecteurs de Jacobi, avec un centre de masse *R* et un hyperangle ϑ.

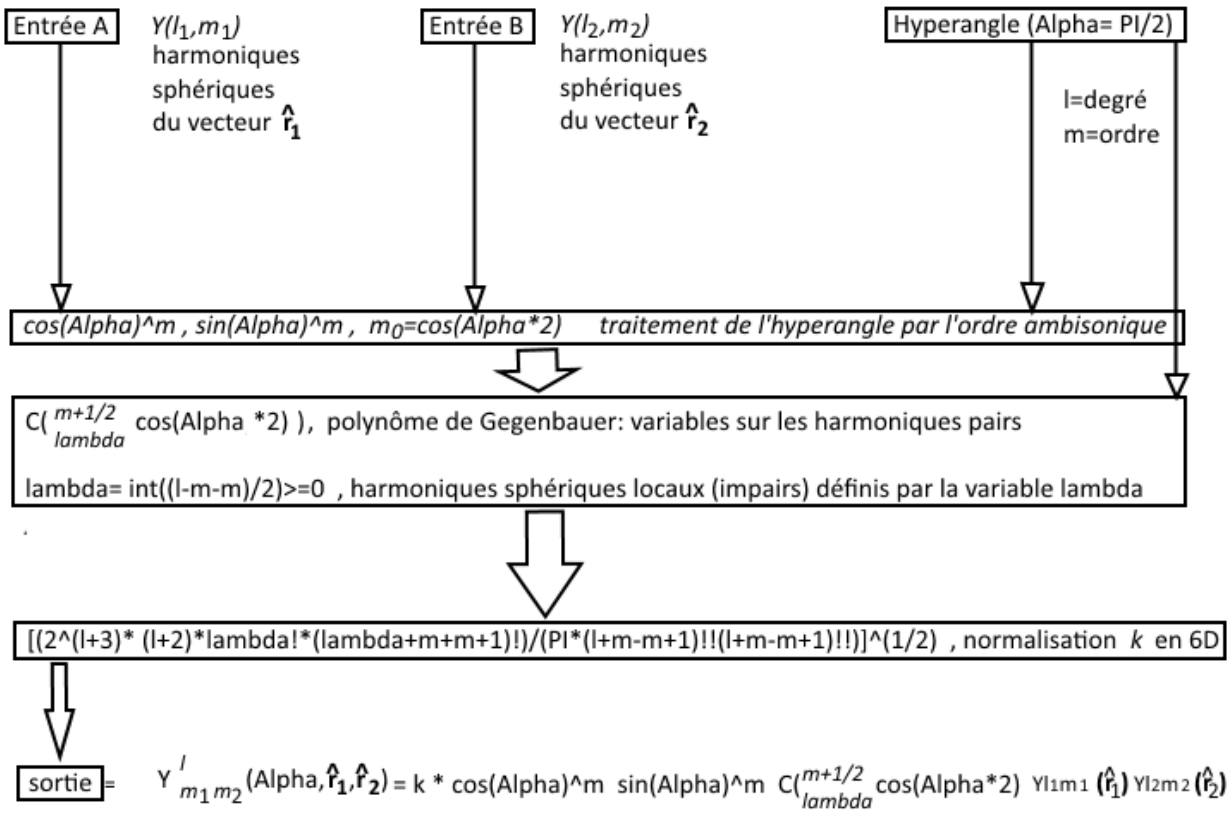
Pour construire des espaces sonores paradoxaux, j'associe les deux vecteurs aux deux sources du même son. Le centre de masse est une coordonnée commune aux deux vecteurs et désigne l'espace depuis lequel les deux sources sonores sont calculées pour être entendues. J'associe donc l'espace du centre de masse au *sweet-spot*. Quand l'espace n'est pas paradoxal, j'associe le centre de masse au centre spatial commun aux deux sources sonores. Quand l'espace sonore est paradoxal, j'associe le centre de masse à un corps silencieux. C'est donc par variation de l'hyperangle que va s'opérer le fondu-enchaîné entre deux sources distinctes. Plusieurs constructions d'harmoniques hypersphériques sont possibles selon les vecteurs additionnés: 1D+3D, 2D+2D, 3D+3D, 2D+2D+2D, je choisis d'encoder les modèles 2D+2D et 3D+3D (fig.4).

3.2 Choix du polynôme dans l'encodage des harmoniques hypersphériques.

L'hyperangle traite non seulement les deux vecteurs, mais aussi leur centre de masse. Dans les harmoniques sphériques, le polynôme de Legendre module l'azimut selon l'élévation. Dans les harmoniques hypersphériques, le polynôme de Jacobi, lui, module les deux vecteurs selon les variations hyperangulaires du centre de masse. Mon choix de composition musicale consiste à ce que les deux sources sonores soient toujours symétriques, alors que le polynôme de Jacobi pourrait prendre en charge des cas asymétriques en modulant l'amplitude du canal ambisonique selon deux ordres possiblement différents, ce qui augmenterait le nombre de canaux, avec neuf ordres pour le premier degré et vingt-cinq pour le second. Dans le domaine hypersphérique l'augmentation du nombre de canaux ambisoniques est au carré du domaine sphérique.
Au delà du second degré, le polynôme de Jacobi est sujet à différentes interprétations, mais aucun atome exotique à plus de trois orbitales n'a pour l'instant été observé (une orbitale atomique se décompose également en harmoniques sphériques)[15][7]. Cette augmentation de l'ordre ambisonique semble impliquer une opération de décodage en 6D ; or le décodage, même s'il est irrégulier (en simulant des haut-parleurs virtuels), doit permettre une restitution dans l'espace tridimensionnel où nous vivons. J'ai donc décidé de remplacer pour l'instant le polynôme de Jacobi par le polynôme de Gegenbauer : ce

[14] Fuks, Benjamin ; « Résolution de l'équation de Schrödinger à 3 corps par la méthode des coordonnées hypersphériques », Laboratoire de Physique Subatomique et de Cosmologie de Grenoble, 5 Octobre 2004.

[15] Feynman, Richard ; Leighton, Robert; Sands, Matthew, « Mécanique quantique », Dunod, 2018. Chapitre 19 « L'atome d'hydrogène et le tableau périodique », p.432-439, «19.6 Le tableau périodique ».



(fig.4) Shémas d'implémentation des harmoniques hypersphériques en 6D.

polynôme, dit ultrasphérique, est une extension du polynôme de Legendre vers les dimensions supérieures et constitue un cas particulier du polynôme de Jacobi où les deux ordres sont les mêmes[16].

Dans la construction des HSH, le degré du polynôme est exprimé selon une variable λ, par une opération dont seuls les résultats entiers positifs sont considérés comme non nuls. Les harmoniques où λ est égal à zéro sont nommés harmoniques sphériques locaux (*Zonal Spherical Harmonics*, ou ZSH) [17]. Ces harmoniques locaux sont donc traités par l'azimut, l'élévation et l'hyperangle, mais ne répondent pas aux variables polynomiales données au centre de masse par l'hyperangle. Le polynôme de Gegenbauer ne module que les harmoniques dans les degrés ambisoniques pairs, ceux-là mêmes qui disparaissent par annulation de phase quand l'espace devient paradoxal dans le sens de l'azimut.

4. RÉACTIONS DE LA COMMUNAUTÉ D'ÉCOUTE

La qualité opératoire du procédé d'espace sonore paradoxal dépend avant tout du contexte musical composé par la superposition d'espaces sonores paradoxaux défilant à différentes vitesses. Je compose ces espaces sonores principalement pour la communauté d'écoute de *free-party* à laquelle j'appartiens. Cette communauté comprend elle-même des habitués et des néophytes, qui vont personnellement interpréter la composition de l'espace sonore. La plupart des teufeurs découvrent l'ambisonie en *free-party* alors qu'ils sont habitués à la monophonie statique qui est une norme générale. Mon traitement spatial du *kick* régulier, matériau primordial en Tekno, consiste à créer une enveloppe hyper angulaire synchronisée sur les variations azimutales, elles-mêmes synchronisées sur la pulsation: modulé par l'hyperangle, un tour complet de l'azimut devient un demi-tour qui se répète à l'identique pour

[16] Stein, Elias ; Weiss, Guido, *Introduction to Fourier Analysis on Euclidean Spaces* (https://archive.org/details/introductiontofo0000stei), Princeton (New Jersey), Princeton University Press, ISBN 978-0-691-08078-9, 1971

[17] Suetin, P.K. (2001) [1994], "Ultraspherical polynomials" (https://www.encyclopediaofmath.or g/index.php?title=U/u095030), in Hazewinkel, Michiel (ed.), *Encyclopedia of Mathematics*, Springer Science+Business Media B.V. / Kluwer Academic Publishers, ISBN 978-1-55608- 010-4.

chaque *kick*. L'enveloppe indique le début et la fin du paradoxe spatial sur chaque demi-tour. Je peux alors moduler le paradoxe spatial en décorrellant l'hyperangle, l'azimut, et la pulsation, pour mettre en valeur l'attaque ou le déclin du *kick* selon la direction et la vitesse désirées.

Depuis, et même un peu avant l'introduction des espaces sonores paradoxaux dans mon *live*, j'ai observé différentes réactions quand je jouais sur un dispositif ambisonique en *free-party*: alors que l'écoute habituelle pousse à danser face à une source sonore monophonique, chacun cherche une nouvelle position d'écoute qui lui convient quand les haut-parleurs sont disposés en ambisonie.

Le sens dans lequel défilent les flux spatialisés pourrait influencer ce choix individuel du positionnement d'écoute, mais la position où je choisis de jouer sur le *dance-floor* influence de manière plus manifeste la position d'écoute du public, qui peut me prendre pour repère visuel. Les personnes avec qui je m'entretiens à propos des conditions d'écoute en ambisonie peuvent choisir de se placer en fonction du *sweet-spot*. Je prends en compte ces divers choix d'écoute quand je compose.

CONCLUSION

En 6D, le traitement du son par les harmoniques hypersphériques se rapproche du traitement des harmoniques sphériques par l'optimisation *inPhase,* qui rend les sources sonores plus directionnelles en réduisant les contributions des harmoniques dans les ordres ambisoniques supérieurs selon des valeurs constantes. Ce traitement a également pour effet d'élargir le champ du *sweet-spot* à l'échelle du *dance-floor*, ce qui assure plus de confort d'écoute. Ce résultat vient de l'intervention de deux opérations successives de normalisations dans la construction des HSH 6D. Une seule normalisation intervient en 4D, où l'optimisation 2D peut alors servir à élargir le *sweet-spot,* notamment en rééquilibrant les harmoniques sphériques pairs et impairs.

Je développe actuellement une configuration avec six sources sonores réparties en trois types de traitements hypersphériques. Cette configuration actuelle fonctionne jusqu'au cinquième ordre ambisonique en 4D, et jusqu'au troisième ordre en 6D, à une fréquence d'échantillonnage de 48kHz.

L'implémentation de l'encodage hypersphérique en Faust permet de comprendre cette limite de l'ordre ambisonique: l'opération fait intervenir des produits de factorielles qui augmentent avec l'ordre ambisonique et deviennent vite des valeurs trop hautes à calculer pour le processeur. L'implémentation des factorielles dans la bibliothèque d'objets mathématiques de Faust pourrait permettre d'utiliser les encodages hypersphériques à des ordres ambisoniques plus élevés.

RÉFÉRENCES

[1] Grynszpan, Emmanuel; « Bruyante Techno », Mélanie, 1999.

[2] Picciola, Daniel; « Le *live* en free-party: contexte culturel, approche du temps, commentaires d'écoute et analyse », CICM, Université de Paris 8, 2015.

[3] Colafrancesco, Julien; « Spatialisation de sources auditives étendues. Applications musicales avec la bibliothèque HOA », CICM, Université de Paris 8, 2015.

[4] Daniel, Jérome; « Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia », LAM, Université de paris 6, 2001.

[5] Guillot, Pierre; « La représentation intermédiaire et abstraite de l'espace comme outil de spatialisation du son. Enjeux et conséquences de l'appropriation musicale de l'ambisonie et des expérimentations dans le domaine des harmoniques sphériques. », CICM, Université de Paris 8, 2017.

[6] Meremianin, A.V., « Hyper-spherical harmonics with arbitrary arguments », Journal of mathematical physics *50*J, 013526, 2009, Department of Theoretical Physics, Voronezh State Universuty, 394006, Vorozneh, Russia.

[7] Feynman, Richard ; Leighton, Robert; Sands, Matthew, « Mécanique quantique », Dunod, 2018.

LES INSTRUMENTS DE MUSIQUE NUMERIQUES POUR LES PERSONNES EN SITUATION DE HANDICAP COGNITIF ET MENTAL : EXEMPLES D'APPLICATIONS VIA LE WEB

Matteo Olivo
Laboratoire ECLLA - Université de Saint-Etienne/Université de Lyon
matteo.olivo@univ-st-etienne.fr

RÉSUMÉ

L'article aborde le sujet des instruments de musique numériques pour les personnes en situation de handicap cognitif et mental. Dans la première partie je passerai en revue les principaux types d'instruments de musique numériques adressés à toutes les formes de handicap, et les définitions utilisées dans la littérature pour les identifier. Cette analyse du contexte me permettra de mieux situer et décrire, ensuite, les applications que j'ai développées dans le cadre de mon doctorat au sein du département de musicologie de l'Université Jean Monnet de Saint-Etienne, codirigé au laboratoire Parcours Santé Systémique de l'Université Lyon 1. La recherche concerne la conception de programmes pédagogiques et d'instruments de musique numériques pour le développement des compétences psychosociales et musicales des personnes en situation de handicap cognitif et mental[1]. En particulier, je décrirai les adaptations apportées pour rendre les outils accessibles, le processus de design suivi et l'implémentation informatique de deux applications. Enfin, j'analyserai les résultats des expérimentations menées et je définirai les perspectives futures de la recherche.

1. LES INSTRUMENTS DE MUSIQUE NUMERIQUES INCLUSIFS

1.1 Etat de la recherche et terminologie utilisée

Depuis des décennies, les technologies musicales et audiovisuelles sont de plus en plus diffusées dans le domaine du handicap de toutes formes (moteur, mental, cognitif, sensoriel et psychique). En effet, ces dispositifs numériques permettent de créer des instruments de musique adaptés aux besoins et capacités de l'utilisateur, en démocratisant ainsi l'accès à la pratique musicale et en ouvrant de nouvelles possibilités expressives. En raison de leur accessibilité et adaptabilité, ces outils trouvent une large variété d'applications : dans des contextes pédagogiques formels et informels [16], dans le cadre de performances musicales de niveau amateur et avancé[2] [9], et aussi au sein de parcours thérapeutiques, comme par exemple dans les programmes de musicothérapie ou d'autres initiatives visant à améliorer les capacités cognitives, motrices ou le bien-être des personnes en situation de fragilité [2][8].

Si dans la littérature concernant l'informatique musicale le thème des instruments de musique pour les personnes en situation de handicap n'a pas encore été beaucoup exploré [6], on peut toutefois constater un intérêt croissant pour le développement et l'analyse de ce type de dispositifs de la part des institutions de recherche, ainsi que des entreprises et des organisations caritatives[3]. En effet, grâce au protocole MIDI, qui a permis une transmission rapide et simple des commandes musicales, à partir des années 1980 beaucoup d'instruments de musique « adaptés » ont été conçus, comme par exemple le *Soundbeam*[4], dispositif basé sur un faisceau sonore[5] qui déclenche des événements MIDI lorsqu'il est coupé, ou le *Magic Flute*[6], instrument qui permet de jouer des notes MIDI à l'aide d'un capteur de pression respiratoire. Ils représentent des exemples des nombreuses possibilités créatives que ce protocole de communication a ouvert dans ce domaine [6]. L'attention croissante pour ce type d'instruments est mise en évidence par le fait que le sujet est de plus en plus abordé dans le cadre d'importantes conférences du domaine de l'informatique musicale, comme par exemple la *NIME (New Interfaces for Musical Expression)* et la SMC (*Sound and Music Computing)*. De plus, des conférences spécifiques sur ce thème sont apparues, comme la *OHMI - Music Making for People with Disability*, entièrement dédiée aux instruments de musique pour les personnes ayant un

[1] Les différentes catégories de handicap ont été définies par la loi n. 2005-102 du 11/02/2005, intitulée *« Loi pour l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées »*.

[2] Mc Pherson, Morreale et Harrison inscrivent les instruments utilisés dans ce contexte dans la catégorie des « *performance-focused instruments »*.

[3] Voir, à ce propos, les organisations *Drake Music* (www.drakemusic.org, site consulté le 27/11/2021), *Open Up Music* (openupmusic.org, site consulté le 27/11/2021) ou *EyeHarp association* (eyeharp.org, site consulté le 3/12/2021).

[4] www.soundbeam.co.uk [consulté le 2/4/2022].

[5] Il s'agit d'un sonar, technologie née pour permettre la détection du mouvement à grande distance sous l'eau.

[6] www.mybreathmymusic.com/en/glenn-2 [consulté le 2/4/2022].

handicap moteur. Cet événement prévoit aussi un prix pour les développeurs de dispositifs de ce type.

Selon Emma Frid [4], qui a effectué un travail de recherche à ce sujet, basé sur l'analyse de 113 documents publiés entre 1990 et 2018[7], il n'y a pas encore une terminologie unique, communément reconnue par la communauté scientifique, concernant les instruments de musique inclusifs. Dans la publication issue de cette recherche, Frid utilise l'expression *Accessible Digital Musical Instruments* (ADMIs), dont elle fournit la définition suivante : *« Accessibles musical control interfaces used in electronic music, inclusive music practice and music therapy settings »* [5]. L'expression *Accessible Instruments* est aussi utilisée par d'autres auteurs, parmi lesquels Harrison et McPherson [7]. Le terme *adaptive*, qui met en valeur la capacité de ces dispositifs d'être personnalisés en fonction des besoins de l'usager, apparait aussi souvent dans la littérature du domaine, avec des tournures différentes : par exemple, Graham-Knight et Tzanetakis [6] utilisent l'expression *Adaptive Music Technology*, en définissant cette notion de la manière suivante : *« The use of digital technologies to allow a person who cannot otherwise play a traditional musical instrument to play music unaided ».* Par contre, Vamvakousis et Ramirez [19] utilisent plutôt la formulation *Adaptive Digital Musical Instruments*. Enfin, dans certaines publications scientifiques du domaine, on trouve d'autres expressions comme *Inclusive music* et *Assistive music technology* [13].

1.2 Catégories de dispositifs existantes, technologies employées

Plusieurs auteurs ont essayé d'élaborer une classification des instruments de musique adressés aux personnes en situation de handicap. Parmi les travaux que j'ai pu examiner à ce sujet, les recherches de Frid mentionnées ci-dessus [4] me semblent être l'une des analyses les plus vastes du point de vue de la taille et de la variété de l'échantillon examiné, et ainsi fournir une image assez représentative de l'état de l'art actuel dans ce domaine : en effet, 83 instruments différents sont étudiés dans ce travail, concernant plusieurs formes de handicap (physique, cognitif, mental et sensoriel). La plupart des autres recherches publiées, par contre, ne concerne que les instruments pour des personnes en situation de handicap moteur (voir, par exemple, [6]). La classification a été effectuée sur la base du type d'interface et du mode principal d'interaction musicale de l'instrument. Les catégories recensées sont les suivantes :

- contrôleurs tangibles : objets physiques qui peuvent être touchés ou mis en mouvement, pour produire du son (voir, par exemple, l'instrument *WamBam*) ;
- contrôleurs sans contact : dispositifs qui utilisent des capteurs ou des caméras pour détecter les mouvements du corps dans l'espace ; les mouvements ne doivent pas impliquer le fait de toucher activement des objets (comme par exemple dans le cas du *SoundBeam*) ;
- instruments adaptés : instruments existants qui ont étés modifiés ou augmentés, comme par exemple les *hyperinstruments* (voir, à ce propos, la basse adaptée analysée par Harrison et McPherson [7]) ;
- interfaces musicales *Brain-Computer* : dispositifs qui utilisent des capteurs biométriques comme les électrodes pour monitorer l'électroencéphalographie, les EEG, etc. (voir, par exemple, le dispositif *Biomuse*) ;
- contrôleurs du regard : instruments qui utilisent le système de *eye-tracking* (comme le *EyeHarp*, analysé par Vamvakousis et Ramirez [19]) ;
- contrôleurs à écran tactile ;
- contrôleurs audio : instruments qui utilisent le microphone comme source sonore ;
- interfaces actionnées par la bouche : contrôleurs de vent qui utilisent des capteurs de respiration (comme le *Magic flute* mentionné ci-dessus) ;
- contrôleurs portables et prothèses : dispositifs qui peuvent être portés ou utilisés comme une extension du corps du musicien (voir, par exemple, le *HipDisk*) ;
- interfaces contrôlées par la souris de l'ordinateur (voir, à ce propos, le dispositif *MidGrid*).

Parmi les 83 instruments analysés, 39 produisent une réponse monomodale (principalement sonore), 40 bimodale (auditive-visuelle dans 33 cas et auditive-vibrotactile dans 7 cas) et 4 trimodale (auditive-visuelle-vibrotactile). Les types de capteurs les plus utilisés pour le développement de ces instruments sont les capteurs tactiles, les accéléromètres et les caméras. Les instruments recensés sont souvent réalisés en utilisant comme base des circuits créés avec des microcontrôleurs, ou d'autres technologies commerciales comme la Nintendo *Wii-mote*, la *Kinect* ou le capteur *Leap Motion*. Concernant les actionneurs, les plus utilisés sont les solénoïdes, les *faders* motorisés et les actionneurs de vibration. En général, les développeurs cherchent des technologies bon marché, mais accessibles et fiables.

2. LES APPLICATIONS MUSICALES CONÇUES DANS LE CADRE DE MON DOCTORAT DE RECHERCHE

2.1 Types d'applications développées et objectifs pédagogiques

L'objectif du projet de recherche est de concevoir des parcours pédagogiques, à l'aide d'outils numériques, visant au développement des compétences psychosociales (confiance en soi, communication verbale et non verbale, capacité de gérer les relations au sein d'un groupe, etc.) et musicales des personnes en situation de handicap cognitif et mental ; les déficits sensoriels ne sont pas analysés dans le cadre de la recherche et les handicaps moteurs ne sont visés que lorsqu'ils sont

[7] Il s'agit d'actes des conférences NIME, SMC et *ICMC - International Computer Music Conference*, articles scientifiques, ouvrages et thèses de doctorat.

associés à des formes de déficience cognitive ou mentale (la dyspraxie, dont je parlerai dans la suite de l'article, est l'unique forme de handicap moteur prise en compte). Le projet s'adresse principalement aux jeunes (élèves du collège, adolescents en dehors du cadre scolaire), mais aussi aux adultes. Les compétences psychosociales sont des capacités dont l'acquisition contribue au bien-être et à la santé[8] de la personne, et se fait principalement à travers des activités de groupe : la musique, en tant que pratique collective qui facilite la création de relations, peut représenter un véhicule assez efficace à ce propos. Les applications que je développe pour la réalisation de ces parcours doivent, donc, être accessibles et intuitives pour des personnes ayant des difficultés du point de vue cognitif et n'ayant pas d'expérience dans le domaine musical ; ces outils doivent permettre aux participants au projet de jouer immédiatement, sans passer par une acquisition graduelle de la maitrise de l'instrument, pour créer une dynamique de groupe et garder la concentration [6]. Mc Pherson, Morreale et Harrison inscrivent ce type d'applications dans la catégorie des *therapeutic instruments*, qu'ils définissent de la manière suivante :

> *Accessible instruments […] designed to elicit the therapeutic or wellbeing aspects of music making for disabled people with physical and cognitive impairments and learning difficulties, who may be identified as 'non-musicians'. Here we refer to the latter case as 'therapeutic instruments', referring to the design goals of enabling musicking for wellbeing purposes, and not necessarily solely for use in formal music therapy.* [9].

Le choix de centrer la composition sur le thème du paysage sonore naturel est cohérent avec l'objectif décrit ci-dessus : en effet, travailler dans le domaine du son, et pas des notes, permet d'éviter de rentrer dans des aspects trop techniques et de passer immédiatement à une phase de création. Les applications développées simulent (en utilisant des échantillons enregistrés ou des techniques de synthèse sonore[9]) des bruits de la nature ou le son de certains instruments de musique (comme par exemple le djembé ou le didgeridoo) qui, dans l'imaginaire collectif, peuvent être associés à ce type d'environnement sonore. Même les simulations des instruments restent très intuitives et permettent de produire du son facilement.

En plus des compétences psychosociales, le projet vise (surtout pour les activités du projet qui se déroulent en milieu scolaire) à développer des compétences théoriques, en cohérence avec les programmes ministériels d'éducation musicale du cycle 3ème et 4ème : par exemple, l'apprentissage de certaines notions de base comme celles de fréquence, amplitude, rythme, la capacité à distinguer le timbre et les catégories principales des instruments, et d'autres. Pour poursuivre ces objectifs je suis en train de développer une deuxième série d'outils numériques, avec la collaboration des professionnels de l'Education Nationale.

Olivier Nicolas a élaboré une classification des applications pédagogiques à utiliser dans le cadre de l'enseignement musical : 1) les applications en dehors du domaine musical, mais dont les fonctionnalités servent un but d'apprentissage (par exemple un outil pour faciliter la lecture des partitions aux personnes dyslexiques) ; 2) les applications musicales d'entraînement pur, catégorie subdivisée en deux sous-catégories : les exerciseurs, plus centrés sur le contenu et ayant moins de distractions sonores et visuelles, et les *serious games*, plus ludiques et animés, et basés sur un système de niveaux qui rend la progression plus attractive ; 3) les applications pour l'interprétation, l'improvisation et la création [10]. En suivant cette classification, nous pouvons inscrire la première série d'applications numériques susmentionnée, visant à la réalisation de la composition autour du paysage sonore, dans la catégorie des outils pour l'interprétation, l'improvisation et la création ; les autres applications pédagogiques, pour l'apprentissage des notions musicales de base, appartiennent à la catégorie des outils pour l'entraînement pur, et notamment des exerciseurs.

2.2 Technologies et environnements de programmation utilisés

Les applications[10] sont développées principalement sous forme d'applications web en utilisant la Web Audio Api, une Api Javascript de haut niveau, pour le traitement de l'audio à baisse latence et la synthèse sonore dans l'environnement web[11]. Le processus de traitement du signal audio a lieu au sein d'un système modulaire (*AudioContext*) conçu par le *World Wide Web Consortium* (W3C) sous forme de graphe audio. Dans cet environnement, des nœuds audio (*AudioNodes*), ayant la fonction de producteurs, transformateurs ou analyseurs du signal, sont connectés entre eux et branchés ensuite à la sortie de l'application. La Web Audio Api fournit une série de nœuds audio natifs qui implémentent les fonctions les plus communes (lecteurs de *buffers*, filtres, spatialisation du son, convolution, etc.) [3]. Cependant, pour pouvoir créer des algorithmes de traitement du signal plus complexes, il est aussi possible de développer des nœuds personnalisés (*AudioWorkletNodes*)[12], codés en Javascript directement ou dans le langage *WebAssembly*. Ce dernier peut être généré par des *Domaines Specific Languages* (*DSL*), comme par exemple Faust. Faust (*Functional Audio Stream*) est un langage fonctionnel, synchrone, spécifiquement conçu par le Grame (Centre National de Création Musicale à

[8] Selon la définition fournie par l'Organisation Mondiale de Santé, la santé est conçue de manière holistique, c'est-à-dire comme un état de bien-être physique, mental et social, et pas uniquement comme l'absence de maladie.

[9] Comme dans le cas de l'application « La mer et les vagues », décrite dans le paragraphe 3.1.

[10] https://www.matteoolivo.com/.

[11] https://www.w3.org/TR/webaudio/#introductory [consulté le 23/12/2021].

[12] https://developers.google.com/web/updates/2017/12/audio-worklet [consulté le 23/12/2021].

Lyon) pour le traitement audionumérique en temps réel et la synthèse sonore [3]. Le compilateur de ce langage permet, en utilisant le *transpiler* Emscripten, basé sur la technologie *LLVM - Low Level Virtual Machine*, de traduire le code Faust directement dans le langage *WebAssembly*, à partir du fichier dsp[13] [11]. Une fois le répertoire où se trouve le fichier dsp sélectionné sur le terminal, il suffit d'écrire le script suivant :

```
faust2wasm -worklet osc.dsp
```

Cette commande génère un fichier *WebAssembly* en format binaire (osc.wasm), et un fichier Javascript (*wrapper*) nommé osc.js, qui permet de transformer le code dsp en nœud audio, qui pourra être intégré dans le script Javascript de l'application et connecté à d'autres nœuds[14]. Le nom du fichier dsp est utilisé pour définir le constructeur du *AudioWorkletNode* (Faustosc). La fonction *startosc* décrite ci-dessous permet de créer le nœud, récupérer la description JSON de la UI et les parcours pour accéder aux paramètres dsp, et enfin de connecter le nœud à la sortie audio de la machine/dispositif. La méthode *setParamValue* assigne la valeur 0.6 au paramètre *freq*. La modification des paramètres peut être aussi contrôlée de manière dynamique dans le script Javascript, à travers un gestionnaire d'événements associé à un élément de l'interface web (un *slider*, un bouton, etc.).

```
function startosc(){
  var pluginURL = ".";
  var plugin = new
Faustosc(audioContext, pluginURL);
  plugin.load().then(node => {
  osc_dsp = node;
  console.log(osc_dsp.getJSON());
  console.log(osc_dsp.getParams());
osc_dsp.connect(audioContext.destination);
  });
}
node.setParamValue("/osc/freq",0.6);
```

Pour certaines des applications web que j'ai développées, j'ai utilisé uniquement les nœuds natifs de la Web Audio Api, tandis que pour d'autres j'ai créé des *AudioWorkletNodes* avec Faust, en suivant la procédure précédemment décrite. Cela m'a permis d'utiliser les librairies (par exemple la librairie sur la synthèse par modèles physiques) et les nombreuses fonctionnalités que cet environnement met à disposition, optimisées pour le traitement du signal audio. En plus de la Web Audio Api, j'ai aussi utilisé d'autres Api Javascript comme la Api *Sensor* (notamment le constructeur *Accelerometer*), pour récupérer les valeurs de l'accéléromètre[15], et *MediaStream Recording*[16] pour enregistrer le son capté par le microphone des dispositifs. Enfin, en ce qui concerne le développement des interfaces, j'ai utilisé les langages traditionnels de la programmation web (HTML et CSS) et le *framework* Bootstrap qui permet de créer assez facilement des interfaces réactives, afin de pouvoir utiliser l'application sur différents types de supports.

Parallèlement aux applications web, j'ai aussi développé d'autres prototypes d'applications en Max/MSP[17]. En effet, par rapport à la Web Audio Api, qui a été créée plus récemment[18], Max offre plusieurs avantages : d'un côté, ayant un plus long historique et une plus grande communauté de développeurs, il possède plus de fonctionnalités et permet de créer des outils plus complexes. De l'autre côté, Max/MSP est basé sur un système de programmation par objets graphiques qui, par rapport à la programmation textuelle de la Web Audio Api, est plus intuitif et permet de développer des prototypes plus rapidement. Une fois testés et validés, ces outils seront développés sous forme d'applications web.

2.3 Les adaptations apportées pour rendre les applications plus accessibles

Généralement, dans la littérature concernant les sciences de l'éducation, l'intégration des technologies numériques dans l'enseignement est citée comme un véhicule pour développer des compétences-clés, comme la pensée computationnelle ou la littératie en matière de médias [1]. Dans le cadre de ma recherche, le choix d'utiliser des applications à la place des instruments de musique traditionnels poursuit principalement un autre objectif, qui consiste à rendre ces outils pédagogiques plus accessibles et adaptables aux capacités cognitives et physiques des usagers. Le développement des compétences numériques représente un objectif secondaire du projet : en effet, les activités menées jusqu'à présent ont montré que la plupart des jeunes impliqués maitrisent déjà assez bien les dispositifs numériques et sont capables d'en comprendre rapidement le fonctionnement, malgré leurs difficultés cognitives[19].

Comme indiqué ci-dessus, le projet s'adresse aux personnes en situation de handicap mental ou cognitif, c'est-à-dire des individus ayant un développement intellectuel et une capacité d'apprentissage significativement inférieures à la moyenne, ou un dysfonctionnement des autres fonctions cognitives : l'attention, la mémoire, le langage, la capacité perceptive (gnosies), la capacité à exécuter des séquences coordonnées de gestes (praxies), la cognition sociale ; les troubles spécifiques de l'apprentissage (dyslexie, dyspraxie, dyscalculie, etc.) s'inscrivent également dans la catégorie du handicap cognitif. Compte tenu de

[13] *Emscripten* génère le code *WebAssembly* à partir du code C++.

[14] Voir, à ce propos, https://github.com/grame-cncm/faust/tree/master-dev/architecture/webaudio [consulté le 23/12/2021].

[15] https://developer.mozilla.org/en-US/docs/Web/API/Accelerometer [consulté le 24/12/2021].

[16] https://developer.mozilla.org/en-US/docs/Web/API/MediaStream_Recording_API [consulté le 24/12/2021].

[17] https://www.matteoolivo.com/maxapplications_en.html

[18] Le premier *working draft* de la Web Audio Api date du 25/11/2011 (https://www.w3.org/TR/2011/WD-webaudio-20111215/, page consultée le 24/12/2021), tandis que la première version de Max/MSP, qui intégrait la partie dédiée au traitement du signal (*Musical Signal Processing*), a été développée en 1997 par la société *Cycling74*.

[19] Voir, à ce propos, le chapitre 4.

l'ampleur du public ciblé, il est très difficile de trouver des solutions transversales, qui peuvent faire face à tous les troubles mentionnés. Les adaptations apportées aux applications, donc, visent à compenser les déficits que l'on retrouve le plus souvent dans ce public, et qui peuvent impacter sur l'utilisation de ce type d'outils : principalement, les difficultés ciblées sont celles liées à la compréhension (dues au déficit mental), à la lecture, à la motricité fine et à la coordination des mouvements. Les aménagements apportés, décrits ci-dessous, ont été identifiés sur la base de l'analyse de la littérature du domaine, de certains documents publiés par le Ministère de l'Education Nationale sur le site Eduscol[20] et des ressources en ligne de la *WAI – Web Accessibility Initiative*[21], mises à disposition par le W3C et concernant le développement et le design des sites web. Une confrontation avec plusieurs professionnels du domaine éducatif et médico-social a été également effectuée.

- Aménagements pour faciliter la compréhension du fonctionnement des applications : pour rendre les logiciels plus accessibles, la quantité de fonctionnalités et d'éléments graphiques sur l'interface a été réduite au minimum. De plus, les interfaces contiennent des informations écrites assez détaillées pour permettre aux utilisateurs, et aussi à leurs parents et aux professionnels qui les accompagnent, d'en comprendre facilement le fonctionnement. Enfin, le texte sur les interfaces est entièrement écrit en français, en évitant au maximum le vocabulaire technique, et sur les boutons des petites icônes ont été placées, qui permettent de mieux comprendre leur fonction (par ex. des icônes représentant le type de son déclenché). Le changement de couleur des boutons, lorsqu'ils sont activés, permet d'avoir un retour visuel de l'interaction effectuée. En ce qui concerne les applications web, l'accessibilité est encore plus marquée, étant donné que les applications ne nécessitent aucune installation de logiciels : il suffit de cliquer sur un site internet pour pouvoir y accéder.

- Aménagements pour faire face aux troubles liés à la lecture : tous les textes présents sur les interfaces ont été écrits avec la police *OpenDyslexic*, conçue pour faciliter la lecture aux personnes dyslexiques[22]. En plus, d'autres adaptations ont été apportées concernant la taille de la police, l'utilisation du caractère gras, l'interligne, le contraste des couleurs. Les explications concernant le fonctionnement des applications sont réalisées aussi en format audio, à travers une technique de synthèse vocale[23] basée sur l'utilisation de l'API Javascript *Web Speech*[24] ; la vitesse de lecture peut être réglée en fonction des besoins de l'usager et la longueur du discours est réduite au minimum, de manière à fournir les informations essentielles et de permettre, en même temps, la compréhension du message. En effet, la courte durée du texte permet d'éviter l'information transitoire, un phénomène qui a des effets négatifs sur l'apprentissage et qui se produit lorsqu'une modalité de présentation permanente (par exemple un texte écrit) est transformée en présentation transitoire (par exemple sous forme orale) [17]. Les mêmes explications sont aussi fournies sous forme textuelle, au sein d'un document qui s'affiche lorsqu'un bouton sur l'interface est pressé : ce texte est rédigé avec des couleurs différentes pour permettre de distinguer les syllabes, et une ligne sur deux est surlignée en couleur[25]. Ces aménagements ont été apportés pour faciliter la lecture aux personnes dyslexiques et dyspraxiques ; la dyspraxie entraine également des déficits visuo-spatiaux et peut provoquer des difficultés concernant la lecture[26].

- Aménagements pour faire face aux difficultés liées à la motricité fine et à la coordination des mouvements : la série d'instruments de musique numériques pour la composition autour du paysage sonore est composée d'applications pour le web et d'autres prototypes développés en Max/MSP. Ces derniers peuvent être contrôlés par des outils différents : des contrôleurs MIDI avec des *faders*, des *pads*, un clavier, la souris de l'ordinateur. De plus, en ce qui concerne les applications web, certains paramètres du son sont contrôlés par l'accéléromètre du dispositif : cela permet de les modifier en temps réel, de manière intuitive et à travers des gestes simples ; dans ces outils l'interface est réactive, de façon à pouvoir les utiliser sur plusieurs types de supports : ordinateur portable et fixe, tablette, *smartphone*. Cette variété d'outils permet à chaque utilisateur de choisir le type d'application, le support et le mode d'interaction les plus adaptés, en fonction des propres capacités motrices ; quel que soit le dispositif utilisé, le mode d'interaction reste toujours très simple et intuitif. L'adaptabilité à plusieurs types de supports permet aussi de réduire les inégalités concernant l'accès aux technologies numériques : en effet, il suffit d'avoir un *smartphone* et une connexion web, pour pouvoir jouer de la musique partout.

2.4 Le processus de design des applications et la méthode de recherche appliquée

Le projet a prévu la mise en place d'un système d'évaluation portant sur trois domaines : le développement des compétences psychosociales, l'acquisition des compétences musicales (objectif concernant principalement les activités en milieu scolaire), le niveau d'accessibilité des dispositifs numériques réalisés et leur impact sur la réalisation du projet. L'évaluation est basée sur la méthode qualitative et, notamment, sur l'observation participante des groupes

[20] https://eduscol.education.fr/ [consulté le 26/12/2021].

[21] https://www.w3.org/WAI/ [consulté le 26/12/2021].

[22] https://opendyslexic.org/ [consulté le 24/12/2021].

[23] Au moment de la rédaction de l'article, seulement certaines applications ont été équipées du dispositif de la synthèse vocale, parmi lesquelles l'application « Djembé » : https://www.matteoolivo.com/app_djembeseq_fr.html.

[24] https://developer.mozilla.org/fr/docs/Web/API/Web_Speech_API [consulté le 28/03/2022].

[25] Les boutons sont visibles sur la figure 1 (paragraphe 3.2).

[26] Voir, à ce propos, le site du projet « Le cartable fantastique » : www.cartablefantastique.fr [consulté le 28/03/2022].

pendant les activités pédagogiques, et sur la réalisation d'entretiens individuels à la fin du parcours avec les participants au projet, leurs parents (quand ils sont disponibles) ou les professionnels qui les accompagnent. Des captations vidéo sont aussi effectuées durant les activités musicales.

Pendant la première année de la recherche (2020/2021) deux parcours pédagogiques ont été réalisés, avec deux groupes de personnes en situation de handicap mental et cognitif suivies par des associations : le premier était composé de quatre adolescents (deux filles et deux garçons) ayant le syndrome de *Williams et Beuren*. Il s'agit d'une maladie d'origine génétique, caractérisée par une déficience intellectuelle de légère à modérée, des capacités motrices et un rapport à l'espace problématiques, un comportement hyper-sociable et, souvent, une aptitude pour la musique et un sens du rythme assez développé. Le deuxième parcours a été réalisé avec un groupe de quatre adultes (trois hommes et une femme) ayant plusieurs types de handicap cognitif : il s'agissait principalement de déficits importants liés à la mémoire et au langage oral. Un participant avait aussi les membres inférieurs, et une partie des membres supérieurs, paralysés. Ces premières expérimentations ont permis de tester, et d'apporter des améliorations aux prototypes des applications réalisés : en effet, en cohérence avec la plupart des projets dans ce domaine, le design des instruments de musique numériques a suivi un processus itératif et participatif, dans le cadre duquel les utilisateurs ont joué un rôle central [4][20]. Les participants ont testé plusieurs types d'outils, et leurs réactions ont été observées à chaque étape du processus de développement des prototypes, comme indiqué dans les recommandations élaborées par Graham-Knight et Tzanetakis [6]. Les entretiens qualitatifs, ainsi que l'échange constant avec les parents et les professionnels des associations partenaires, ont permis d'intégrer les données collectées à travers l'observation.

3. L'IMPLEMENTATION INFORMATIQUE DES APPLICATIONS : EXEMPLES D'APPLICATIONS WEB DEVELOPPEES

3.1 L'application « La mer et les vagues »

Cette application simule les bruits de la mer et des vagues, reproduits en utilisant la technique de la synthèse soustractive et notamment en filtrant un bruit rose et un bruit blanc, une méthode classique dans le domaine de la synthèse de textures sonores [15]. Le déclenchement des vagues peut se faire avec le mouvement du dispositif, à travers le *mapping* de l'accéléromètre. En effet, l'application a été conçue pour des dispositifs portables (*smartphone*, tablette) et vise à sensibiliser l'utilisateur au geste instrumental. Pour ceux qui ne peuvent pas profiter de cette fonctionnalité, à cause de difficultés liées à la coordination motrice, il est quand même possible de déclencher les vagues en appuyant sur un bouton.

En ce qui concerne le développement de l'application, le processus de traitement du signal a été codé en Faust : le bruit des vagues est créé par un générateur de bruit rose (généré en utilisant la fonction standard de Faust *no.pink_noise*), filtré avec un filtre *butterworth* passe-haut (la fonction standard de Faust *fi.highpass*). Le signal est multiplié par un générateur d'enveloppe (fonction *en.adsr*) dont la *release* (rel), qui représente la longueur de la vague, est contrôlée par un *horizontal slider*. La fonction primitive *select2* de Faust permet de choisir le mode de déclenchement de la vague : en appuyant sur le bouton, ou en utilisant le mouvement du dispositif. Le fonctionnement de ce deuxième mode est basé sur la fonction *gate_acc*, qui lisse la valeur de l'accéléromètre (récupérée dans le code Javascript, comme expliqué après), et ensuite la transforme en une valeur comprise entre 0 et 1 pour pouvoir déclencher et arrêter le son.

```
rel = hslider ("LENGTH_WAVE", 3.5, 2.0, 6.0,
0.001) : si.smoo;
envelope = en.adsr
(0.1,0.01,0.9,rel,gate_wave)*gain;
s = hslider("SELECTOR_GATE",1,0,1,1);
gate_wave = gate_acc, gate_but : select2(s);
acc_x = hslider("XROTATION",0,-100,100,0.001);
gate_acc = +((abs(acc_x + acc_x'))> 100) ~
*(0.99) : min(1) : max(0);
process = (sea_synth * gain2) / 2,
       (wave_synth * envelope : echo) / 2;
```

Le bruit de la mer est simulé de manière analogue à celui de la vague, avec un générateur de bruit blanc (fonction *no.noise*) filtré par un filtre *butterworth* passe-bas (fonction *fi.lowpass*). Dans la fonction *process* les deux sources sonores sont mises en parallèle et leur amplitude est divisée par 2, pour éviter la saturation du signal. Un effet écho codé en Faust est ajouté aux vagues.

En suivant la procédure précédemment décrite, le code Faust est transformé en nœud de l'*Audio Context* et ensuite connecté à la sortie de la machine. Dans le script Javascript, un *slider* (élément HTML <input>) gère le contrôle de la longueur de la vague, à travers un gestionnaire d'événements associé ; les volumes de la vague et de la mer sont contrôlés de la même manière. Enfin, un bouton permet de sélectionner le mode de déclenchement de la vague : s'il est activé, la valeur du paramètre "SELECTOR_GATE" du code Faust mentionné ci-dessus, récupérée avec la méthode *setParamValue* du nœud, est égal à 0, et le démarrage par l'accéléromètre est sélectionnée ; en revanche, lorsque le bouton est désactivé, le déclenchement de la vague se fait en cliquant sur un autre bouton placé sur l'interface web.

```
const lengthwaveControl =
document.getElementById('length');
lengthwaveControl.addEventListener('input',
function() {
merapp_dsp.setParamValue("/merapp/LENGTH_WAVE",
parseFloat(event.target.value));
}, false);
val = 0;
merapp_dsp.setParamValue("/merapp/SELECTOR_GATE
", val);
```

Dans le code Javascript, la valeur de l'accéléromètre (mouvement autour de l'axe des x) est récupérée en utilisant le constructeur *Accelerometer* de la Api *Sensor* ;

la lecture est faite 60 fois par seconde. En utilisant la fonction *convertRange*, les valeurs récupérées sont ensuite converties en valeurs comprises entre -100 et 100 (le min. et max. du paramètre XROTATION dans le code Faust), et visualisées graphiquement au sein d'un *slider* (l'élément ayant comme id 'acc') présent sur l'interface.

```
function convertRange(value, r1, r2) {
        return (value-r1[0])*(r2[1]-
        r2[0])/(r1[1]-r1[0]) + r2[0];
}
let acl = new Accelerometer({frequency: 60});
acl.addEventListener('reading', ()=> {
var accX = acl.x;
var accScale = convertRange(accX,
[-10,10], [-100,100]);
merapp_dsp.setParamValue("/merapp/XROTATION",
parseFloat(accScale));
const triggerControl =
document.getElementById('acc');
triggerControl.value = accScale ;
});
```

3.2 Le « Djembé séquenceur »

Il s'agit d'une application qui simule les trois types de sons principaux d'un djembé, c'est-à-dire la basse, le son tonique et le son claqué (ou *slap*). Un *step séquencer*, composé de trois lignes de boutons qui déclenchent les sons mentionnés, permet de programmer des *patterns* rythmiques, de les répéter en boucle et les modifier en temps réel. Il s'agit d'une interface accessible et intuitive, à travers laquelle les utilisateurs peuvent visualiser et comprendre assez facilement la structure interne des *patterns* rythmiques. La vitesse et les volumes des trois sons sont contrôlés par des *sliders*. J'ai développé cette application en suivant un modèle illustré sur le site *MDN Web Docs*[27], dont j'ai modifié l'interface, pour la rendre plus accessible, et le processus de traitement du signal[28]. En effet, contrairement au modèle, qui est entièrement codé en Javascript, le dsp de mon application est développé en Faust, à partir de la fonction *pm.djembe* qui reproduit le timbre du djembé en utilisant la synthèse par modèles physiques. Dans le code Faust les trois sons mentionnés sont déclenchés par des boutons (à leur tour activés par les boutons présents sur l'interface web de l'application), et leur volume est contrôlé par un *hslider*. Les paramètres de chaque son (fréquence de la fondamentale, *strikeposition* et *strikesharpness*) ont été déterminés de façon empirique, de manière à pouvoir bien distinguer les trois sons lorsque l'application est utilisée sur des dispositifs (*smartphones*, ordinateurs portables) ayant des enceintes non professionnelles de petites dimensions. Dans la fonction *process* les trois sources sonores sont mises en parallèle. Le dsp est enfin transformé en nœud dans le script Javascript.

```
gatebass = button("gatebass");
gainbass = hslider("Volumebass", 0.95, 0.0,
1.0, 0.01);
bass = vgroup ("bass", gatebass : pm.djembe(70,
0.1, 0.9, 1) * gainbass);
```

Le mécanisme à la base du séquenceur a été codé en Javascript et s'appuie sur la propriété *currentTime,* qui récupère la durée en secondes du temps écoulé après la création de l'*AudioContext.* Cette propriété, qui est basée sur le *audio clock* interne de l'ordinateur/dispositif, permet de programmer donc une suite d'événements sonores synchronisée avec cet horodatage ; grâce au niveau de précision très élevé de *currentTime* (la propriété restitue une valeur arrondie au 15ème chiffre décimal), cette suite d'événements sonores reste cohérente même pendant de longs intervalles de temps. Le fonctionnement du séquenceur est basé sur un système de planification qui permet de prévoir en avance à quels moments les sons seront joués ; les événements sonores détectés et planifiés sont ajoutés, avec la méthode *push,* à une « file d'attente » représentée par l'*array* stocké dans la variable *notesInQueue*. Le changement de la valeur de l'attribut *aria-checked* des *pads* (boutons) du séquenceur, modifie la valeur des paramètres *gate*[29] du dsp, et déclenche les sons.

```
const notesInQueue = [];
function scheduleNote(beatNumber,time {
notesInQueue.push({note:beatNumber,time:time});
if(pads[0].querySelectorAll('button')[beatNumbe
r].getAttribute('aria-checked') === 'true') {
djembeseq_dsp.setParamValue("/djembeseq/bass/ga
tebass",1);
} else if
(pads[0].querySelectorAll('button')[beatNumber]
.getAttribute('aria-checked') === 'false') {
djembeseq_dsp.setParamValue("/djembeseq/bass/ga
tebass", 0);
// On répète la même structure pour le son tonique
et claqué
}
```

Chaque ligne du séquenceur est composée de 8 boutons ; 1 bouton correspond à 1 *beat* donc, chaque ligne, en 4/4, équivaut à 2 mesures. Ce choix représente un point d'équilibre entre l'accessibilité de l'interface et la possibilité de créer des variations sur les patterns rythmiques : plus de mesures auraient permis plus de variations, mais auraient obligé à réduire la taille des boutons, en rendant l'application difficile à utiliser sur des petits écrans. A chaque *beat,* la fonction *nextNote* indiquée ci-dessous fait avancer le *timing* interne du séquenceur d'un *beat* (1 bouton), en repartant à zéro lorsqu'il arrive à 8. La variable tempo correspond à la vitesse en bpm du séquenceur par défaut, *currentNote* est le numéro de *beat* (et de bouton) initial et *nextNoteTime* l'intervalle temporel après lequel le prochain événement sonore doit avoir lieu (par défaut fixé à 0) ; enfin, la variable *secondsPerBeat* calcule la durée d'un *beat*.

[27] https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Advanced_techniques [consulté le 06/01/2022].

[28] Le *framework tone.js* comprend des classes qui peuvent être utilisées pour la création d'un séquenceur ; cependant, j'ai décidé de ne pas m'en servir et de développer le séquenceur à partir de zéro, pour améliorer mes connaissances dans le domaine de la programmation web.

[29] Il s'agit plus précisément des boutons *gatebass*, *gatetone* et *gateslap* du dsp, qui déclenchent les trois sons du djembé.

```
let currentNote = 0;
let nextNoteTime = 0.0;
let tempo = 60.0;
function nextNote() {
  const secondsPerBeat = 60.0 / tempo;
  nextNoteTime += secondsPerBeat;
  currentNote++;
  if (currentNote === 8){
      currentNote = 0;
  }
}
const lookahead = 25.0;//(millisecondes)
const scheduleAheadTime = 0.1;//(secondes)
function scheduler() {
 while(nextNoteTime<audioCtx.currentTime +
scheduleAheadTime){
 scheduleNote(currentNote,nextNoteTime);
 nextNote();
 }
 timerID =
window.setTimeout(scheduler,lookahead);
}
```

La variable *scheduleAheadTime* indique combien de secondes le système va vérifier à l'avance s'il y a des événements sonores programmés ; ensuite, la fonction *scheduler* vérifie si un événement sonore est prévu durant cet intervalle de temps, calculé à partir de la valeur de *currentTime.* Si tel est le cas, les fonctions *scheduleNote* et *nextNote* sont appelées. Enfin, la propriété *setTimeout* exécute la fonction *scheduler* selon la fréquence temporelle déterminée par *lookahead*. Le démarrage du séquenceur se fait à l'aide d'un bouton placé en haut de l'interface : quand il est activé, la fonction *scheduler* est appelée et la valeur initiale de *nextNote* est égale à celle relevée par la propriété *currentTime*. Le bouton déclenche aussi une autre fonction, basée sur la méthode *requestAnimationFrame*, qui permet de visualiser l'avancement du curseur du séquenceur sur les *pads*. Lorsque le bouton est désactivé, la méthode *clearTimeout* de *window* annule le comptage démarré par la propriété *window.setTimeout*, et celle-ci cesse d'être appelée.

Figure 1. Interface de l'application.

4. L'ANALYSE DES RESULTATS DES EXPERIMENTATIONS MENEES

Les deux parcours pédagogiques réalisés jusqu'à présent[30] permettent de tracer des premières conclusions par rapport à l'impact, et au niveau de l'accessibilité, des outils développés. Ces expérimentations se sont déroulées pendant la première année de la recherche ; les dispositifs testés étaient, donc, encore à un stade non avancé : il s'agissait de prototypes développés en Max/MSP et d'applications web contenant seulement les fonctions basiques, sans la plupart des aménagements décrits au paragraphe 2.3. D'autres dispositifs comme le *Gramophone*[31], développé par le Grame, et le capteur de mouvement *Leap Motion*[32] ont été utilisés pour mettre à disposition des participants un éventail d'outils numériques et de modes d'interaction différents, afin de comprendre leurs capacités et préférences.

En ce qui concerne l'impact de ces technologies sur la réalisation du projet, trois questions ont été posées à chaque participant lors de l'entretien, visant à comprendre si les dispositifs numériques avaient facilité la réalisation et l'exécution de la composition, quel avait été le support fourni et quel type d'instruments de musique les participants auraient-ils préférés, entre les technologies numériques et les instruments de musique traditionnels. Toutes les personnes interviewées des deux groupes (six sur six) ont affirmé que les technologies ont fortement facilité la réalisation de la composition musicale ; selon deux personnes, celle-ci ne pouvait pas être réalisée avec des instruments de musique traditionnels, compte tenu du niveau des participants, qui étaient des débutants ou des personnes ayant une expérience musicale assez limitée. Cinq personnes sur six ont affirmé avoir préféré jouer avec des instruments numériques, plutôt que traditionnels. En ce qui concerne le type de support fourni par les technologies, dans deux cas la question n'a pas été posée pour des raisons liées à la longueur de l'entretien, et les autres réponses étaient pour la plupart hors sujet. Seulement une personne du deuxième groupe, présentant des déficits cognitifs et physiques assez importants, a fourni des éléments pertinents en affirmant que les instruments de musique numériques permettent de produire du son plus rapidement, et que grâce à l'ordinateur, il a pu jouer d'une seule main. Les parents des participants, présents seulement dans le cas du premier groupe, ont tous confirmé l'importance fondamentale des technologies pour atteindre le résultat final.

En ce qui concerne l'accessibilité des outils, l'observation participante a permis d'évaluer principalement deux domaines : la capacité des participants à interagir avec les applications et leurs interfaces, et la capacité à comprendre les fonctionnalités des outils et à savoir se repérer au sein des environnements numériques utilisés. En ce qui concerne le premier, les deux groupes qui ont participés aux

[30] Voir le paragraphe 2.4.

[31] https://www.amstramgrame.fr/gramophone/about/.

[32] https://www.ultraleap.com/product/leap-motion-controller/

parcours ont montré des résultats assez analogues. En général une bonne maitrise des outils a été constatée, et une capacité suffisante à interagir avec les éléments des interfaces. La taille des écrans a constitué un aspect névralgique. Les *smartphones*, par exemple, ont donné lieu dans certains cas à des difficultés d'interaction : les *sliders* présents au sein des applications web étaient souvent difficiles à déplacer sur les téléphones portables ; en particulier, le déplacement par étapes, comme celui d'un curseur qui sélectionne la note de l'instrument, s'est avéré particulièrement compliqué (par rapport au déplacement continu). Au cours du projet, la taille des *sliders* a été agrandie, en améliorant de manière significative l'utilisation de ces éléments. Une deuxième considération concerne le mode d'interaction, et notamment la présence d'un dispositif *hardware*, externe ou intégré avec le software : les applications qui prévoyaient ce type de dispositifs sont apparues plus attractives et intuitives par rapport à celles contrôlées à travers la souris de l'ordinateur ou l'écran tactile. Les applications commandées par des contrôleurs MIDI (il s'agit des applications développées en Max/MSP), notamment ceux caractérisés par la présence de *pads* ou de *faders*, ont été toujours choisies par les participants lorsqu'il y en avait la possibilité. En plus, l'utilisation de *faders* pour contrôler les *sliders* s'est avérée plus efficace et précise par rapport à la souris ou au déplacement du doigt sur l'écran. De façon générale, dans les cas où la dimension gestuelle est valorisée davantage, l'interaction devient plus intéressante. Cet aspect a été remarqué aussi par rapport au *Gramophone*, qui, parmi les différents dispositifs, est apparu l'un de plus attractifs pour les participants. Par contre, au-delà d'un certain seuil de complexité du geste, ce type de dispositifs peut facilement se transformer en obstacle : par exemple, avec le *Gramophone*, le réglage du volume à travers le potentiomètre rotatif a posé des problèmes, lorsqu'il devait être effectué pendant que l'appareil bougeait dans l'espace pour moduler le son en fonction de l'accéléromètre. En effet, la coordination des deux mouvements demande un niveau de dextérité supérieur à celui de la plupart des participants. Pour la même raison, l'utilisation du capteur *Leap Motion* s'est avérée complexe : le contrôle des paramètres sonores et le déclenchement des sons avec le mouvement des mains, demandent un niveau de motricité trop avancé pour les participants. Une dernière considération valable pour les deux groupes concerne la dimension visuelle de l'interaction avec les éléments de l'interface : il est très important d'avoir ce type de retour pour que l'utilisateur puisse prendre conscience des interactions effectuées. Un exemple est représenté par les boutons sur les interfaces web : en effet, tous ces éléments ne déclenchent pas du son lorsqu'ils sont pressés, et même dans le cas où cette interaction produit un retour auditif, il peut être couvert par le son des autres dispositifs utilisés dans la salle. En plus, il a été observé que le retour visuel impacte de manière positive sur la concentration de l'utilisateur. Pour ces raisons, un changement de couleur de tous les boutons, lorsqu'ils sont pressés, a été mis en place au cours du projet.

En ce qui concerne la capacité à comprendre les fonctionnalités des outils, et à savoir se repérer au sein des environnements numériques, l'observation des deux groupes a produit des résultats différents. Les adolescents atteints du syndrome de *Williams et Beuren* ont montré une certaine aisance avec la navigation web et, plus en général, avec tous les environnements numériques utilisés. Lorsqu'ils ne comprenaient pas, ils essayaient quand même d'interagir avec les dispositifs, poussés par la curiosité. Leurs parents ont affirmé qu'ils ont l'habitude de se servir du *smartphone* et de l'ordinateur dans leur quotidienneté, pour naviguer sur internet ou utiliser des logiciels d'usage commun. La compréhension des fonctionnalités des applications a été assez immédiate, surtout en ce qui concerne les applications web qui, par rapport à celles développées en Max/MSP, disposent d'interfaces plus simples. Par contre, des difficultés ont été constatées par rapport au deuxième groupe, composé de personnes adultes ayant jusqu'à 55 ans. Bien qu'il s'agisse de personnes ayant des conditions intellectuelles normales (leurs troubles concernaient surtout la mémoire et le langage), dans certains cas même la compréhension des fonctionnalités de base des outils a posé des problèmes : par exemple, deux personnes n'arrivaient pas à trouver le bouton pour le démarrage des applications développés en Max/MSP et souvent, s'ils n'étaient pas suivis, ils s'arrêtaient devant l'écran sans effectuer aucune interaction. Un témoignage intéressant des diverses approches par rapport aux technologies numériques qui caractérisent les différentes générations, ainsi que de l'écart existant du point de vue de la maitrise de ces dispositifs.

5. CONCLUSIONS ET PERSPECTIVES FUTURES DE LA RECHERCHE

La recherche prévoit la réalisation d'une troisième expérimentation pendant l'année 2022/2023, adressée à des élèves du collège en situation de handicap cognitif et mental. Ce public, en effet, constitue la cible principale du projet et pourra bénéficier d'une plus grande variété d'outils numériques, à un stade de développement plus avancé ; ceux-ci contiendront, en effet, tous les aménagements du point de vue de l'accessibilité indiqués dans le paragraphe 2.3. Les résultats des évaluations effectuées semblent valider le choix de mettre à disposition des participants une palette assez large d'outils, de manière à que chaque membre du groupe puisse trouver le dispositif, et le mode d'interaction qui lui convient le plus. Pour cela, la poursuite du travail sur les applications web sera accompagnée par la mise à jour et l'amélioration des prototypes développées en Max/MSP ; cela permettra aussi de pouvoir disposer d'outils adaptés aux différents contextes de jeu. En effet, si les applications équipées d'un contrôleur MIDI, comme celles réalisées en Max/MSP, sont plus attractives, elles sont moins adaptées lorsque l'enfant doit utiliser ces dispositifs à la maison ou à l'école, sans la

présence d'une personne experte. Au sein de ces contextes, les applications web, ubiquitaires et plus accessibles, représentent des solutions plus efficaces. Par rapport à tous ces aspects, la confrontation avec les professionnels et les parents a eu un rôle fondamental, et sera encore plus mise en valeur dans le futur.

Le système d'évaluation sera également amélioré, notamment en ce qui concerne la structuration de l'entretien et la formulation des questions ; l'évaluation portera aussi sur l'acquisition des compétences musicales théoriques, un aspect qui jusqu'à aujourd'hui a été moins approfondi, auquel sera dédiée une partie de l'observation et une série spécifique d'applications qui, comme indiqué ci-dessus, est en train d'être développée.

REFERENCES

1. Baron G.L., Depover, C., (dir.), « Impact du numérique sur les curricula et les programmes d'études », in *Les effets du numérique sur l'éducation. Regards sur une saga contemporaine*, Presses universitaires du septentrion, Villeneuve d'Ascq, 2019.

2. Burland, K., Magee, W.L., « An exploratory study of the use of electronic music technologies in clinical music therapy », *Nordic Journal of Music Therapy*, Vol. 17, 2008.

3. Denoux, S., Letz, S., Orlarey Y., Fober, D., « Composing a Web of Audio Applications », *Proceedings of the Web Audio Conference*, Paris, France, 2015.

4. Frid, E., « Accessible Digital Musical Instruments – A Review of Musical Interfaces in Inclusive Music Practice », *Multimodal Technologies and Interactions*, Vol. 3 (3), 57, 2020.

5. Frid, E., « Accessible Digital Musical Instruments: A Survey of Inclusive Instruments Presented at the NIME, SMC and ICMC Conferences », *Proceedings of the International Computer Music Conference*, Daegu, South Korea, 2018.

6. Graham-Knight, K., Tzanetakis G., « Adaptive Music Technology: History and Future Perspectives », *Proceedings of the International Conference on Computer Music*, Denton, USA, 2015.

7. Harrison, J., McPherson, A.P., « Adapting the Bass Guitar for One-Handed Playing », *Journal of New Music Research*, Vol. 46, 4, 2017.

8. Larsen, J.V., Overholt, D., Moeslund, T.B., « The Prospects of Musical Instruments for People with Physical Disabilities », *Proceedings of NIME – New Interfaces for Musical Expression*, Brisbane, Australie, 2016.

9. McPherson, A., Morreale F., Harrison J., « Musical Instruments for Novices: Comparing NIME, HCI and Crowdfunding Approaches », Holland, S., Mudd, T.,Wilkie-McKenna, K., McPherson, A., Wanderley, M.M., *New Directions in Music and Human-Computer Interaction*, Springer International Publishing, Cham, Suisse, 2019.

10. Nicolas, O., « Idées d'usage des smartphones, tablettes, web et vidéo pour l'enseignement musical spécialisé », Terrien, P., Deveney, G., *L'intégration du numérique dans l'enseignement*, L'Harmattan, Paris, 2018.

11. Ren, S., Letz, S., Orlarey, Y., Michon, R., Fober, D., Buffa, M., Lebrun, J., « Using Faust DSL to Develop Custom, Sample Accurate DSP Code and Audio Plugins for the Web Browser », *Journal of the Audio Engineering Society*, Vol. 68, 2020.

12. Roads, C., *L'audionumérique : Musique et Informatique*, Dunod, Paris, 3ème édition française, 2016.

13. Samuels, K., « Enabling Creativity: Inclusive Music Interfaces and Practices », *Proceedings of the International Conference on Live Interfaces (ICLI)*, Lisbon, Portugal, 2014.

14. Schafer, R.M, *Le paysage sonore. Le monde comme musique*, Wildproject, Marseille, édition française, 2010.

15. Schwarz, D., « State of the art in sound texture synthesis », *Proceedings of the 14th International Conference on digital audio effects (Dafx-11)*, Paris, France, 2011.

16. Swinglers, T., « Getting better all the time : Using music technology for learners with special needs », *Australian Journal of Music Education*, n. 2, 2009.

17. Tricot, A., *Numérique et apprentissages scolaires : quelles fonctions pédagogiques bénéficient des apports du numérique ?*, Cnesco, Paris, 2020.

18. Turner, W., *Javascript for Sound Artists: Learn to Code with the Web Audio Api*, Routledge, Londres, 2017.

19. Vamvakousis, Z., Ramirez, R., « The EyeHarp: A gaze-controlled digital musical instrument », *Frontiers in Psychologies*, 7, 906, 2016.

20. Ward, A., Woodbury, L., Davis, T., « Design Considerations for Instruments for Users with Complex Needs in SEN Settings », *Proceedings of NIME - New Interfaces for Musical Expression*, Copenhague, Danemark, 2017.

ESTIMATION DE PARAMÈTRES DE RESYNTHÈSE DE SONS D'INSTRUMENTS DE MUSIQUE AVEC DES OUTILS DE MORPHOLOGIE MATHÉMATIQUE

Gonzalo Romero-García, Carlos Agón
STMS, IRCAM, Sorbonne Université,
CNRS, Ministère de la Culture, 75004 Paris
{romero, agonc}@ircam.fr

Isabelle Bloch
Sorbonne Université, CNRS, LIP6, Paris
isabelle.bloch@sorbonne-universite.fr

RÉSUMÉ

Les sons représentés par des spectrogrammes peuvent être considérés comme des images dont la dimension horizontale correspond au temps, et la verticale à la fréquence. Dans cet article, nous proposons d'explorer quelques outils algébriques de la morphologie mathématique, théorie largement développée en analyse et interprétation d'images, pour estimer des paramètres de resynthèse d'un son d'instrument de musique en le modélisant comme une partie harmonique à laquelle est ajouté un bruit blanc filtré. En particulier, nous montrons que des transformations non linéaires visant à détecter des structures saillantes permettent de déduire les paramètres des composantes harmoniques d'un signal. Avec les mêmes outils, nous estimons les paramètres de filtrage pour resynthétiser la partie non-harmonique.

1. INTRODUCTION

La synthèse de sons d'instruments de musique est un problème qui a surgi aux débuts du traitement numérique du signal audio. Les reproducteurs de fichiers MIDI utilisaient des modèles très simples formés par l'addition de sinusoïdes harmoniques. Les amplitudes de ces sinusoïdes étaient estimées au moyen de la transformée de Fourier, mais ne pouvaient pas rendre compte d'une évolution temporelle. Des modèles plus complexes ont ensuite été développés, permettant l'évolution temporelle du son, et reposant le plus souvent sur la transformée de Fourier à court terme, plus connue par son sigle en anglais STFT [1]. C'est le cas, par exemple, dans [7], [1] ou encore [17].

Dans cet article, nous proposons de resynthétiser un signal d'un son d'instrument de musique à partir d'un échantillon de celui-ci. Pour ce faire, nous utilisons le modèle proposé dans [18] en modélisant le signal comme une partie générée par synthèse additive (la partie harmonique) à laquelle s'ajoute une partie générée de manière stochastique (la partie non-harmonique). Dans notre cas, la partie stochastique s'obtient par filtrage d'un bruit blanc. Ce modèle est présenté dans la section 2.

1 . *Short-Time Fourier Transform.*

Il s'agit alors d'estimer des paramètres dans notre échantillon qui nous permettent de synthétiser par ce modèle un son similaire. Pour retrouver ces paramètres, nous utilisons une représentation du signal intermédiaire sous forme de spectrogramme de STFT. Cette représentation, présentée en section 3, peut être vue comme une image du son en échelle de gris et fait apparaître les différents partiels qui composent la partie harmonique, ainsi que les distributions fréquentielles de la partie non-harmonique.

La nouveauté proposée dans cet article est l'estimation des amplitudes des partiels et des filtres à l'aide d'outils de morphologie mathématique. Cette théorie, développée dans la deuxième partie du XX$^{\text{ème}}$ siècle [12, 16], est très utilisée pour traiter, analyser et interpréter des images. Dans la section 4, nous présentons quelques outils que nous utilisons pour trouver les paramètres de synthèse.

En particulier, nous exploitons la structure des spectrogrammes où la partie harmonique apparaît sous forme de lignes horizontales ; grâce à des opérations morphologiques, nous détectons les temps, fréquences et amplitudes associés à chaque partiel que nous utilisons ensuite pour la synthèse additive. D'autres opérations morphologiques nous permettent de filtrer ces parties saillantes pour pouvoir utiliser le résultat comme filtre d'un bruit blanc. Ces processus sont expliqués dans la section 5.

Pour évaluer l'efficacité de notre méthode, nous choisissons un cas favorable (un son de synthétique généré par le modèle expliqué en section 2) et nous montrons que nous pouvons estimer ses paramètres de resynthèse avec une très bonne précision (section 5). Ensuite, dans la section 6, nous appliquons le même procédé à des sons d'instruments de musique issus d'une base de données.

2. MODÈLE DE SIGNAL

L'idée de cet article est de reconstruire un signal donné en estimant des paramètres de synthèse à partir de son spectrogramme. Pour ce faire, nous devons d'abord expliquer comment nous construisons le signal et quels sont les paramètres à estimer. Dans cette section, nous exposons un modèle de synthèse composé d'une partie additive, correspondante à la partie harmonique, et d'une partie soustractive, correspondante à la partie non-harmonique.

Pour vérifier l'efficacité de notre méthode, nous allons construire un signal selon ce modèle qui nous servira de test ; nous attendons de notre méthode qu'elle puisse retrouver les paramètres de synthèse d'un tel signal.

2.1. Partie harmonique

La partie harmonique est créée par synthèse additive de sinusoïdes dont les fréquences et les amplitudes varient dans le temps : comme expliqué dans la première section de [1], nous prenons $N \in \mathbb{N}$ partiels sinusoïdaux de durée $T > 0$ dont la fréquence $\omega_n(t)$ et l'amplitude $a_n(t)$ varient en fonction du temps $t \in [0, T]$. Chaque partiel s_n est donné par la formule

$$s_n(t) = a_n(t) \sin\left(\theta_n(t)\right) , \tag{1}$$

où $\theta_n(t) = 2\pi \int_0^t \omega_n(\tau)\, d\tau$, $t \in [0, T]$, $n \in \{0, ...N-1\}$.

Dans le cas de notre signal de référence, nous prenons des fréquences constantes et multiples d'une fréquence fondamentale f_0 :

$$\forall t \in [0, T],\ \omega_n(t) = f_n := (n+1) f_0 . \tag{2}$$

La fonction de phase s'écrit alors :

$$\theta_n(t) = 2\pi \int_0^t (n+1) f_0\, d\tau = 2\pi (n+1) f_0 t . \tag{3}$$

Les amplitudes de notre signal de référence $a_n(t)$ sont des exponentielles décroissantes :

$$\forall t \in [0, T],\ a_n(t) = A_n e^{-2\pi \delta_n t} \tag{4}$$

où les A_n sont les amplitudes initiales et δ_n les facteurs de décroissance [2] . De plus, nous prenons $A_n = A_0 \frac{1}{(n+1)^2}$ comme amplitudes initiales, avec $A_0 > 0$ et des facteurs de décroissance linéaires en fréquence, *i.e.* $\delta_n = \delta f_n = \delta(n+1) f_0$, où $\delta > 0$ est un facteur qui contrôle la décroissance globale.

En combinant ces paramètres, nous obtenons un signal de référence qui s'écrit :

$$s(t) = \sum_{n=0}^{N-1} s_n(t) \tag{5}$$

$$= \sum_{n=0}^{N-1} A_n e^{-2\pi \delta_n t} \sin\left(\theta_n(t)\right) \tag{6}$$

$$= \sum_{n=0}^{N-1} \frac{A_0}{(n+1)^2} e^{-2\pi \delta (n+1) f_0 t} \sin\left(2\pi (n+1) f_0 t\right) . \tag{7}$$

2.2. Partie non-harmonique

La partie non-harmonique du signal est modélisée de manière soustractive par filtrage d'un bruit blanc. Par la suite, nous utilisons des procédés classiques tels que le filtrage, le fenêtrage ou la recomposition d'un signal par la méthode OLA [3] [6]. Pour une présentation de ces outils, nous proposons comme références [13, 14].

Pour générer notre signal de référence, nous procédons comme suit :

1. Génération d'un bruit blanc b donné par une fonction $t \mapsto b(t)$.
2. Décomposition du bruit en trames $b_n(t)$ de longueur $L > 0$ et d'espacement $H > 0$, *i.e.* :

$$b_n(t) = b(t + nH),\ t \in [0, L] . \tag{8}$$

3. Fenêtrage de ces trames par une fenêtre de Hann $w(t) = \sin^2\left(\frac{\pi n}{N}\right)$.
4. Filtrage des trames par un filtre linéaire qui varie dans le temps Θ.
5. Recomposition du signal par la méthode OLA.

Pour notre signal de référence, le filtre que nous utilisons est un filtre passe-bande dont l'amplitude décroît exponentiellement en temps. La réponse en fréquence du filtre au temps $\tau \in [0, T]$ est donnée par :

$$\Theta_\tau(\omega) = \chi_{[f_{c_1}, f_{c_2}]}(\omega) e^{-2\pi \tau \eta} , \tag{9}$$

où $\chi_{[f_{c_1}, f_{c_2}]}$ est la fonction indicatrice de l'intervalle $[f_{c_1}, f_{c_2}]$, f_{c_1} et f_{c_2} sont les fréquences de coupure et $\eta > 0$ est une constante qui détermine la rapidité de la décroissance de l'amplitude du filtre en temps.

2.3. Génération de notre signal de référence

Pour construire notre signal de référence, nous générons d'une part la partie harmonique et d'autre part la partie non-harmonique et nous les additionnons. Le bruit blanc est généré en choisissant des nombres au hasard entre -1 et 1 de manière uniforme. Les paramètres que nous utilisons pour la partie additive et pour filtrer le bruit sont exposés dans la table 1.

Partie harmonique	
N	16
f_0	220 Hz
A_0	0.1
δ	5×10^{-4}

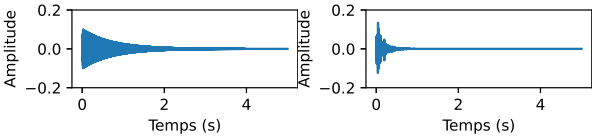
Partie non-harmonique	
L	0.1 s
H	0.01 s
f_{c_1}	100 Hz
f_{c_2}	300 Hz
η	1 Hz

Table 1. Paramètres utilisés pour la synthèse des parties harmonique et non-harmonique de notre signal de référence.

La figure 1 montre le signal de référence. Sa représentation est temporelle et ne nous donne pas beaucoup d'informations sur la composition du signal hormis le fait qu'il a une amplitude décroissante. Dans la section suivante, nous présentons la notion de spectrogramme de STFT qui nous permettra d'avoir une représentation du signal sous forme d'une image où sa composition sera plus évidente.

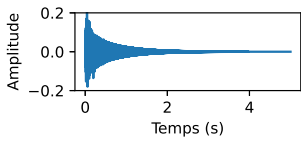
2 . On multiplie les facteurs de décroissance par 2π pour qu'ils soient exprimés en Hz.

3 . *Overlap-add method*.



(a) Partie harmonique

(b) Partie non-harmonique



(c) Somme des parties harmonique et non-harmonique

Figure 1. Signal de référence généré par addition d'une partie harmonique et une partie non-harmonique.

3. SPECTROGRAMMES

Dans cette section, nous présentons les spectrogrammes, considérés comme « images » d'un son. Ce sont sur ces représentations que nous appliquerons les outils morphologiques. Le spectrogramme que nous utilisons est celui de la STFT. Une description détaillée de ces transformations et des fondements mathématiques sur lesquels elles reposent peut être trouvée dans [8].

3.1. Cas continu

La STFT peut être défine dans le cas continu comme suit.

Définition 1 *Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction bornée de classe C^∞. Soit $w : \mathbb{R} \to \mathbb{R}^+$ une fonction de classe C^∞ à support compact. La STFT de f avec fenêtre w est définie par*

$$\begin{array}{rccl} \mathrm{STFT}_w[f] : & \mathbb{R}\times\mathbb{R} & \to & \mathbb{C} \\ & (\tau,\omega) & \mapsto & \int_{\mathbb{R}} f(t)w(t-\tau)e^{-\jmath 2\pi\omega t}\,dt\,. \end{array} \tag{10}$$

La définition de la STFT peut être établie dans plusieurs contextes d'analyse fonctionnelle ; ici, nous utilisons des fonctions infiniment dérivables et une fenêtre à support compact pour garantir l'existence de l'intégrale. Néanmoins, il est fréquent de se placer dans le cadre des fonctions de Schwartz et même de la théorie des distributions. Ces considérations sont explorées dans [8].

Une fois la STFT établie, son spectrogramme se définit comme suit.

Définition 2 *Soient f et w des fonctions comme dans la définition 1. Le* spectrogramme *de STFT de f avec fenêtre w est défini par*

$$\begin{array}{rccl} \mathrm{SPEC}_w[f] : & \mathbb{R}\times\mathbb{R} & \to & \mathbb{R}^+ \\ & (\tau,\omega) & \mapsto & |\mathrm{STFT}_w[f](\tau,\omega)|^2\,. \end{array} \tag{11}$$

Ainsi, un spectrogramme représente l'information d'amplitude de la STFT au carré. On parle souvent de la valeur d'un spectrogramme de f au point (τ,ω) comme de la densité de puissance de f au temps τ et à la fréquence ω. Cette densité de puissance est souvent exprimée en dB par la formule

$$\mathrm{SPEC}^{\mathrm{dB}} = 10\log_{10}(\mathrm{SPEC})\,. \tag{12}$$

Si l'on considère les spectrogrammes en échelle logarithmique, notre ensemble d'arrivée n'est plus $\mathbb{R}^+$ mais $\mathbb{R}\cup\{-\infty\}$; de plus, si l'on considère que $|f|$ est bornée par 1 (ce qui est le cas dans les signaux audio) et que l'intégrale de w est égale à 1, un résultat établi dans [8] affirme que $\mathrm{SPEC}_w[f]$ est borné par 1 et donc que $\mathrm{SPEC}_w^{\mathrm{dB}}[f]$ est borné par 0.

Des précisions sont à faire pour le domaine de la STFT et des spectrogrammes : la première dimension, correspondant au temps, est infinie en théorie mais finie en pratique puisque les signaux d'entrée sont finis. Pour la dimension fréquentielle, même si elle est définie pour tout $\mathbb{R}$ dans la pratique, lorsqu'il s'agit de signaux réels, on ne garde que la partie positive, puisque la partie négative est symétrique hermitienne. De plus, pour un signal échantillonné à une fréquence $F_e \in \mathbb{N}$, nous ne gardons que les fréquences jusqu'à $\frac{F_e}{2}$, rendant la dimension fréquentielle finie.

Ces considérations nous mènent à établir le cadre discret utilisé lors des calculs.

3.2. Cas discret

Pour pouvoir calculer des spectrogrammes dans la pratique, l'entrée est, non pas une fonction continue et infiniment dérivable, mais plutôt un signal échantillonné. La formule de la STFT discrète est donnée dans la définition suivante.

Définition 3 *Soit $(\mathbf{f}[l])_{l=0}^{L}$ un signal de taille $L+1$ échantillonné à une fréquence F_e. Soit $(\mathbf{w}[n])_{n=0}^{N}$ une fenêtre de taille $N+1$. Soit $N_{\mathrm{FFT}} \in \mathbb{N}$, $N_{\mathrm{FFT}} \geq N$ et $H \in \mathbb{N}$. Pour tout $m = 0 : \lfloor \frac{L}{H} \rfloor$, $k = 0 : \lfloor \frac{N_{\mathrm{FFT}}}{2} \rfloor$, on définit la STFT de $\mathbf{f}$ avec fenêtre $\mathbf{w}$ par*

$$\mathrm{STFT}[\mathbf{f}][m,k] = \sum_{n=0}^{N} \mathbf{f}\left[mH + n - \lfloor \frac{N}{2} \rfloor\right] \mathbf{w}[n] e^{-\jmath 2\pi n \frac{k}{N_{\mathrm{FFT}}}} \tag{13}$$

La coordonnée m de la STFT correspond au temps $\frac{mH}{F_e}$ et la coordonnée k correspond à la fréquence $\omega_k = F_e \frac{k}{N_{\mathrm{FFT}}}$. On complète $\mathbf{f}$ par des zéros aux bords pour que la formule soit bien définie.

La table 2 donne les paramètres choisis pour le calcul des spectrogrammes. La fenêtre choisie est la fenêtre de Hann et est normalisée pour que sa somme fasse 1.

Les figures 2 et 3 montrent les spectrogrammes des parties harmonique et non-harmonique de notre signal de référence commençant après 1 s de silence et d'une durée $T = 4$ s, synthétisées avec les paramètres donnés dans la table 1.

F_e	44 100 Hz
N	4410
N_{FFT}	8820
H	441
$\mathbf{w}[n]$	$\sin^2\left(\frac{\pi n}{N}\right)$

Table 2. Paramètres utilisés pour la STFT.

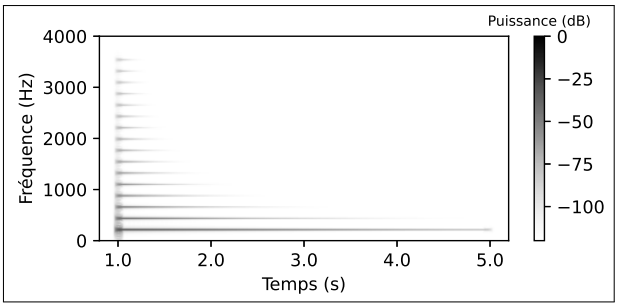


Figure 2. Spectrogramme de la partie harmonique du signal de référence.

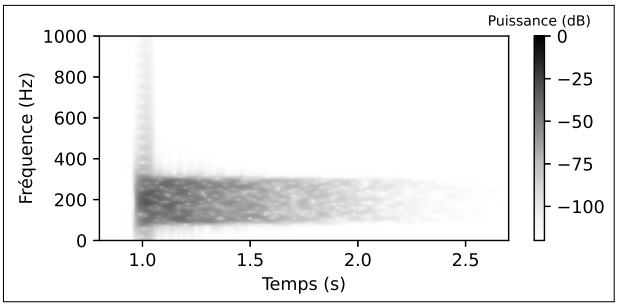


Figure 3. Spectrogramme de la partie non-harmonique du signal de référence.

4. MORPHOLOGIE MATHÉMATIQUE

Nous proposons maintenant d'utiliser quelques outils de la morphologie mathématique pour analyser les spectrogrammes et déduire les paramètres de synthèse des parties harmonique et non-harmonique.

Nous nous plaçons dans le cadre de la morphologie mathématique déterministe définie sur des treillis complets [4]. Nous utilisons ici les opérations de base (dilatation et érosion), le résultat de leurs compositions (ouverture et fermeture) et des opérateurs dérivés (résidus d'ouverture et squelette). Toutes les définitions que nous présentons peuvent être trouvées par exemple dans [3, 9, 15].

4.1. Dilatation et érosion

Définition 4 *Soient* $(L_1, \leq_1)$ *et* $(L_2, \leq_2)$ *deux treillis complets, et* $\vee_i, \wedge_i (i = 1, 2)$ *les supremum et infimum associés. Un opérateur* $\delta : L_1 \to L_2$ *est une* dilatation *si*

$$\forall X_1 \subseteq L_1\,,\ \bigvee\nolimits_2 \delta(X_1) = \delta\left(\bigvee\nolimits_1 X_1\right). \tag{14}$$

Un opérateur $\varepsilon : L_2 \to L_1$ *est une* érosion *si*

$$\forall X_2 \subseteq L_2\,,\ \bigwedge\nolimits_1 \varepsilon(X_2) = \varepsilon\left(\bigwedge\nolimits_2 X_2\right). \tag{15}$$

Ces définitions implicites peuvent prendre des formes particulières explicites, en particulier avec la notion d'*élément structurant*. Nous nous plaçons ici dans le treillis des fonctions dont l'ensemble d'arrivée est un treillis complet.

Proposition 1 *Soit E un ensemble. Soit* $(T, \leq)$ *un treillis complet. Alors, la paire* $(T^E, \leq)$ *où* T^E *est l'ensemble des fonctions* $f : E \to T$ *et* $\leq$ *est la relation d'ordre partiel donnée par*

$$\forall f, g \in T^E, f \leq g \Leftrightarrow \forall p \in E, f(p) \leq g(p) \tag{16}$$

est un treillis complet.

Les opérations de supremum et infimum sont données par : $\forall \{f_i\}_{i\in I} \subseteq T^E$ *(où I est un ensemble d'indices quelconque),*

$$\bigvee_{i\in I} f_i : E \to T, \quad p \mapsto \bigvee_{i\in I} f_i(p) \tag{17}$$

$$\bigwedge_{i\in I} f_i : E \to T, \quad p \mapsto \bigwedge_{i\in I} f_i(p) \tag{18}$$

Nous demandons en outre que $(E, +)$ soit un groupe additif de sorte que l'on puisse parler de translation d'une fonction ; la translation de $f \in T^E$ par l'élément $h \in E$ est définie par

$$T_h f : E \to T, \quad p \mapsto f(p + h)\,. \tag{19}$$

En ce qui concerne T, deux choix sont privilégiés : le premier est le treillis binaire $(\{0, 1\}, \leq)$, ce qui donne lieu à la morphologie binaire. Dans ce cas, les fonctions $\{0, 1\}^E$ peuvent être identifiées aux sous-ensembles de E par la fonction indicatrice, la translation de fonctions devient la translation de sous-ensembles et la relation $\leq$ entre fonctions devient l'inclusion $\subseteq$ entre sous-ensembles. Nous parlons alors du treillis $(\mathcal{P}(E), \subseteq)$. Le deuxième choix est d'utiliser un ensemble numérique comme $\mathbb{R}$, ce qui donne lieu à la morphologie en échelle de gris. Ce dernier choix pose le problème que $(\mathbb{R}, \leq)$ n'est pas un treillis complet, car il n'y a pas d'élément maximal ou minimal. Il est alors habituel de travailler avec l'extension $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$.

Les propositions suivantes montrent des définitions d'érosion et dilatation dans les cas binaire et en échelle de gris.

Proposition 2 *Soit* $(\mathcal{P}(E), \subseteq)$ *le treillis complet des sous-ensembles d'un groupe additif* $(E, +)$. *Soit* $B \in \mathcal{P}(E)$. *Alors, les opérateurs*

$$\begin{array}{rccc} \delta_B : & \mathcal{P}(E) & \to & \mathcal{P}(E) \\ & A & \mapsto & \{a + b \in E : a \in A, b \in B\} \end{array} \tag{20}$$

$$\begin{array}{rccc} \varepsilon_B : & \mathcal{P}(E) & \to & \mathcal{P}(E) \\ & A & \mapsto & \{p \in E : T_p B \subseteq A\} \end{array} \tag{21}$$

4 . Un treillis est un ensemble partiellement ordonné où toutes les paires d'éléments ont un supremum noté $\vee$ et un infimum noté $\wedge$. Un treillis complet est un treillis dont tous les sous-ensembles ont un supremum et un infimum.

sont respectivement une dilatation et une érosion. Le sous-ensemble $B \subseteq E$ est appelé élément structurant. On parle alors de dilatation et d'érosion binaires avec élément structurant *B.*

Proposition 3 *Soit $(\overline{\mathbb{R}}^E, \leq)$ le treillis complet des fonctions d'un groupe additif $(E, +)$ dans $\overline{\mathbb{R}}$. Soit $\sigma \in \overline{\mathbb{R}}^E$. Alors, les opérateurs*

$$\begin{aligned} \delta_\sigma : \overline{\mathbb{R}}^E &\to \overline{\mathbb{R}}^E \\ f &\mapsto \delta_\sigma[f] : E \to \overline{\mathbb{R}} \\ & \qquad p \mapsto \sup_{h \in E} (f(h) + \sigma(p - h)) \end{aligned} \tag{22}$$

$$\begin{aligned} \varepsilon_\sigma : \overline{\mathbb{R}}^E &\to \overline{\mathbb{R}}^E \\ f &\mapsto \varepsilon_\sigma[f] : E \to \overline{\mathbb{R}} \\ & \qquad p \mapsto \inf_{h \in E} (f(h) - \sigma(h - p)) \end{aligned} \tag{23}$$

sont respectivement une dilatation et une érosion. Par convention, $\infty - \infty$ vaut ∞ pour la dilatation et $-\infty$ pour l'érosion. Ici, l'élément structurant est la fonction σ (appelée aussi fonction structurante*). On parle alors de dilatation et d'érosion en échelle de gris avec élément structurant σ.*

Un cas particulier très fréquent se présente lorsque l'élément structurant est égal à zéro dans un sous-ensemble A de E et $-\infty$ ailleurs [5] ; dans ce cas, la dilatation (resp. l'érosion) d'une fonction f en un point $p \in E$ revient à prendre le supremum (resp. l'infimum) de f dans l'ensemble translaté de A par $p : T_p A = \{a + p \in E : a \in A\}$.

Ces deux opérateurs permettent d'en construire d'autres, en particulier par composition.

4.2. Fermeture et ouverture

Nous rappelons que, si $(L, \leq)$ est un treillis, un opérateur $\psi : L \to L$ est dit

— croissant si $\forall X, Y \in L, X \leq Y \Rightarrow \psi(X) \leq \psi(Y)$,
— extensif si $\forall X \in L, X \leq \psi(X)$,
— anti-extensif si $\forall X \in L, \psi(X) \leq X$,
— idempotent si $\psi^2 = \psi$,

Définition 5 *Soit $(L, \leq)$ un treillis. Soit $\psi : L \to L$ un opérateur. On dit que*

1. *ψ est une* fermeture *si ψ est croissante, extensive et idempotente,*
2. *ψ est une* ouverture *si ψ est croissante, anti-extensive et idempotente.*

De par leurs propriétés de croissance et idempotence, les fermetures et les ouvertures sont des *filtres morphologiques*.

5 . Cela revient à prendre comme élément structurant la fonction indicatrice de A, notée χ_A, en échelle logarithmique.

Comme dans la section précédente, nous construisons des cas particuliers de fermetures et d'ouvertures, par composition d'érosions et de dilatations, définies en particulier à partir d'éléments structurants.

Proposition 4 *Soit $(L, \leq)$ un des deux treillis complets présentés dans la section précédente* [6] *. Soit $\beta \in L$. Alors,*

1. *$\varphi_\beta := \varepsilon_\beta \circ \delta_\beta$ es une fermeture et*
2. *$\gamma_\beta := \delta_\beta \circ \varepsilon_\beta$ es une ouverture.*

4.3. Résidus d'ouverture et squelette

Dans cette section, nous présentons deux opérateurs qui nous seront particulièrement utiles : les résidus d'ouverture et le squelette morphologique.

Les résidus d'ouverture, présentés dans la définition suivante, permettent d'extraire des informations sur la saillance d'une image. Nous présentons cet opérateur pour le cas en échelle de gris.

Définition 6 *Soit $(\overline{\mathbb{R}}^E, \leq)$ le treillis complet présenté dans les sections précédentes. Prendre les résidus de l'ouverture γ d'un élément $f \in \overline{\mathbb{R}}^E$ revient à lui appliquer l'opérateur*

$$\mathbb{1} - \gamma \tag{24}$$

où $\mathbb{1}$ est l'opérateur identité.

Étant donné que les ouvertures sont anti-extensives, on a $\forall f \in \overline{\mathbb{R}}^E$,

$$(\mathbb{1} - \gamma)(f) \geq 0\,. \tag{25}$$

Si nous pensons l'ouverture comme une sorte de régularisation vers le bas, prendre les résidus d'ouverture nous permet de récupérer les éléments saillants qui ont été filtrés.

L'autre opérateur que nous proposons, le squelette morphologique, est présenté pour le cas binaire grâce à une formule due à Lantuéjoul [11]. Nous demandons aussi que l'espace de base E soit $\mathbb{Z}^2$ pour avoir la notion de boule centrée en 0 de rayon $n \in \mathbb{N}$; dans $\mathbb{Z}^2$ nous appelons boule centrée en 0 de rayon n l'ensemble $B_n = \{x \in \mathbb{Z}^2 : ||x|| \leq n\}$, où le choix de la norme détermine sa forme.

Définition 7 *Soit $(\mathcal{P}(E), \subseteq)$ le treillis complet des sous-ensembles de E ordonnés par l'inclusion. Soit $X \in \mathcal{P}(E)$. Le squelette de X s'obtient par la formule*

$$S(X) = \bigcup_{n \in \mathbb{N}} (\varepsilon_{B_n}(X) \setminus \gamma_B(\varepsilon_{B_n}(X))) \tag{26}$$

où ε_{B_n} est l'érosion binaire avec comme élément structurant une boule centrée en 0 de rayon n, et γ est l'ouverture élémentaire, c'est-à-dire où l'élément structurant est la boule centrée en 0 de rayon 1.

6 . $(\mathcal{P}(E), \subseteq)$ pour le cas binaire et $(\overline{\mathbb{R}}^E, \leq)$ dans le cas en échelle de gris.

5. ESTIMATION DES PARAMÈTRES DE SYNTHÈSE

Dans cette section, nous montrons comment les opérateurs morphologiques présentés auparavant peuvent être utilisés pour détecter les amplitudes, temps et fréquences des partiels harmoniques et la réponse en fréquence variant dans le temps du filtre.

Pour illustrer l'efficacité de notre méthode, nous prenons le signal de référence que nous avons construit dans la section 2, dont nous recherchons les paramètres de synthèse. Le spectrogramme du signal de référence est présenté dans la figure 4.

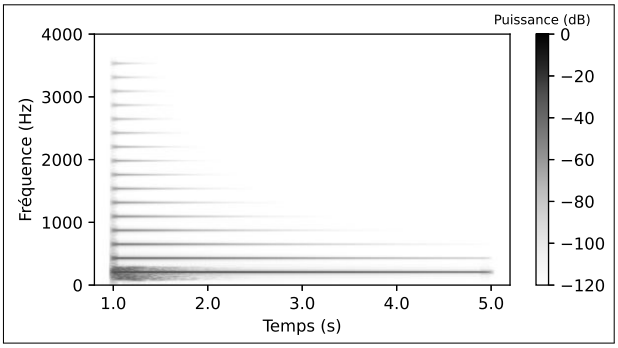


Figure 4. Spectrogramme du signal de référence présenté dans la section 2 et illustré dans la figure 1.

5.1. Estimation des paramètres de la partie harmonique

Pour estimer les paramètres de la partie harmonique, il s'agit d'isoler les lignes horizontales que nous voyons dans la figure 4 en les rétrécissant à des lignes de un pixel d'épaisseur. Pour cela, nous utilisons l'opérateur des résidus d'ouverture ; étant donné que cet opérateur est très sensible au bruit, nous pré-traitons l'image en la filtrant avec une fermeture. Les résidus d'ouverture nous donnent une information de contraste, et nous gardons donc les pixels qui dépassent un certain seuil τ, de sorte à avoir une image binaire. Nous pouvons ensuite calculer le squelette de cette image binaire, et nous obtenons des lignes horizontales d'un pixel d'épaisseur. La donnée des abscisses, ordonnées et intensités de chaque ligne sera considérée comme le résultat de notre processus et nous permettra de synthétiser (après interpolation) un signal qui sera similaire à la partie harmonique du signal d'entrée.

Les éléments structurants présentés par la suite sont définis dans $\mathbb{R} \times \mathbb{R}$ et prennent leurs valeurs dans $\overline{\mathbb{R}}$, par souci de généralité. Pour les calculs, nous réalisons un échantillonnage de ces fonctions en prenant les valeurs aux points de la grille données par le spectrogramme. Souvent, nous prenons des éléments structurants correspondant à des fonctions indicatrices de sous-ensembles A de $\mathbb{R} \times \mathbb{R}$ exprimés en dB : si on appelle σ l'élément structurant, il s'exprime alors comme

$$\sigma = 10 \log_{10}(\chi_A) \,. \tag{27}$$

5.1.1. Fermeture

La première étape consiste à appliquer une fermeture φ_{σ_1} au signal d'entrée, qui permet d'éliminer les variations dues au bruit et à l'étalement spectral. Pour cela, nous prenons comme élément structurant σ_1 la fonction indicatrice du sous-ensemble $A_1 = [0, 0, 05]$ s$\times[0, 50]$ Hz exprimée en dB.

Le résultat de cette opération sur notre signal de référence est montré dans la figure 5.

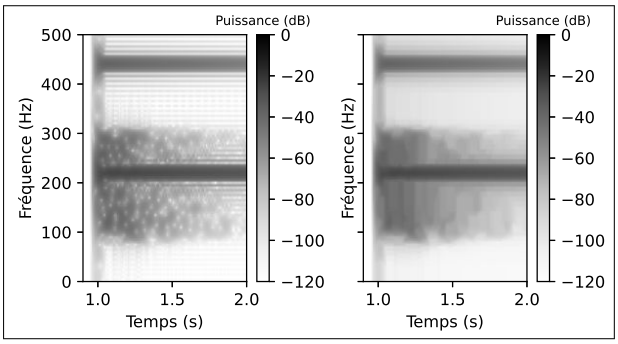


Figure 5. Image d'entrée (à gauche) et le résultat de la fermeture φ_{σ_1} (à droite).

5.1.2. Résidus d'ouverture seuillés

L'étape suivante consiste à prendre les résidus d'ouverture $\mathbb{1} - \gamma_{\sigma_2}$ de la fermeture et à les seuiller à une valeur τ pour obtenir une image binaire. Nous prenons comme élément structurant σ_2 la fonction indicatrice du sous-ensemble $A_2 = \{0\,\text{s}\} \times [0, 25]$ Hz exprimée en dB. Nous utilisons comme seuil $\tau = 3\,\text{dB}$.

Le résultat de cette opération sur notre signal de référence est montré dans la figure 6. Nous voyons comment les lignes horizontales sont isolées du reste.

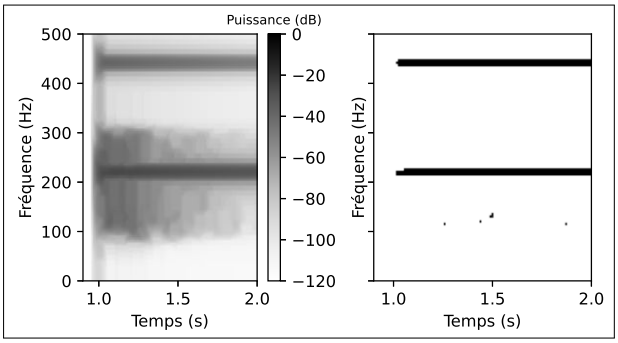


Figure 6. Fermeture (à gauche) et le résultat des résidus d'ouverture $\mathbb{1} - \gamma_{\sigma_2}$ seuillés à 3 dB (à droite).

5.1.3. Squelette morphologique des résidus d'ouverture seuillés

La dernière étape consiste à calculer le squelette morphologique des résidus d'ouverture ; même si les lignes

horizontales des résidus d'ouverture ont déjà isolé les partiels harmoniques, elles ont encore une épaisseur de plusieurs pixels. Nous voulons n'avoir qu'un pixel d'épaisseur et qu'il soit le plus central possible. Pour calculer le squelette, nous utilisons des boules B_n de la pseudo-norme [7] $||(x_1, x_2)|| = |x_2|$, où $(x_1, x_2) \in \mathbb{Z}^2$; ce choix a la particularité qu'elle ne fait rétrécir la forme que dans la direction de la fréquence, propriété qui nous intéresse. Pour la boule centrée en 0 de rayon 1, utilisée dans l'ouverture, nous devons nous restreindre à l'ensemble $\{(0,0),(0,1)\}$ au lieu de l'ensemble $\{(0,-1),(0,0),(0,1)\}$ car sinon nous risquons de faire apparaître des lignes de deux pixels d'épaisseur.

Le squelette morphologique des résidus d'ouverture seuillés est montré dans la figure 7 ; les lignes correspondant aux partiels ont bien été rétrécies jusqu'à une ligne d'un pixel d'épaisseur.

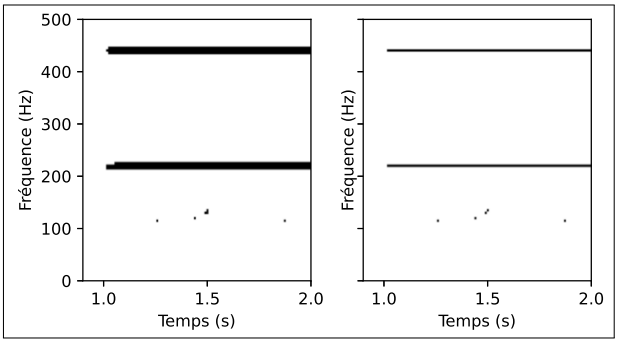


Figure 7. Résidus d'ouverture seuillés (à gauche) et son squelette morphologique (à droite).

5.1.4. Calcul des temps, fréquences et amplitudes des partiels

Finalement, nous nous servons du squelette obtenu pour retrouver les informations des partiels ; pour cela, nous parcourons l'image pixel par pixel et nous parcourons les lignes obtenues en notant à quel temps et à quelle fréquence elles apparaissent. Les amplitudes correspondantes sont les valeurs de l'image fermée [8] .

La figure 8 montre les amplitudes originales correspondant à chacune des fréquences ainsi que le résultat de nos estimations. On constate une parfaite adéquation.

5.2. Estimation des paramètres de la partie non-harmonique

L'estimation des paramètres de la partie non-harmonique est obtenue aussi par des méthodes morphologiques ; nous pré-traitons l'image au moyen d'un fermeture pour limiter l'effet du bruit et de l'étalement spectral (comme dans le cas de la partie harmonique), puis,

7 . Cette définition ne donne pas une norme car elle ne vérifie pas la condition de séparation, mais elle nous sert quand même pour notre cas particulier.

8 . On pourrait argumenter qu'il faudrait prendre celles de l'image originale, mais ce choix est fait pour limiter l'effet du bruit.

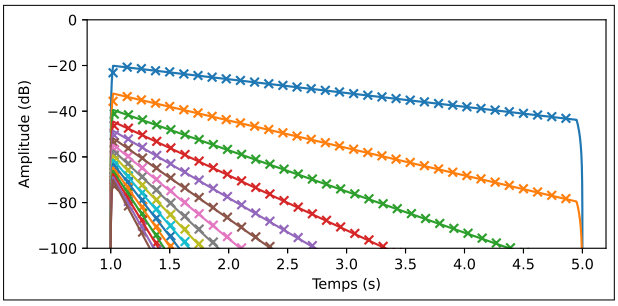


Figure 8. Amplitudes originales (ligne) et nos estimations (croix) ; chaque ligne correspond à une fréquence.

nous appliquons une ouverture pour éliminer les parties saillantes (correspondant aux partiels harmoniques) et garder seulement la partie du spectrogramme correspondant à la partie non-harmonique. Finalement, nous appliquons une érosion pour réduire l'effet de l'étalement spectral.

5.2.1. Fermeture

La fermeture utilisée est la même que dans la section précédente, φ_{σ_1}, et le résultat de son application sur notre image d'entrée est le même que dans la figure 5.

5.2.2. Ouverture

L'ouverture, que nous nommons γ_{σ_3}, a pour élément structurant la fonction indicatrice du sous-ensemble $A_3 = \{0\}$ s $\times$ $[0, 100]$ Hz exprimée en dB. Le résultat sur la fermeture est montré dans la figure 9.

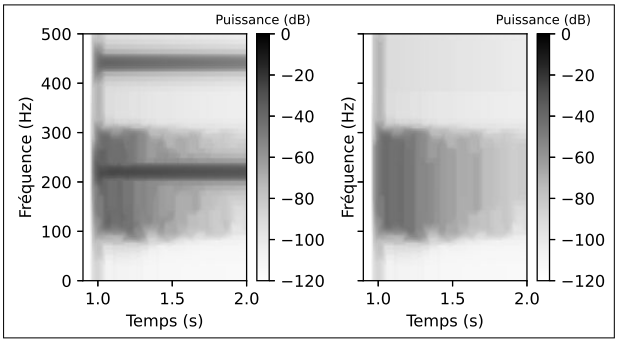


Figure 9. Fermeture (à gauche) et le résultat de l'ouverture γ_{σ_3} (à droite).

5.2.3. Érosion

L'érosion ε_{σ_4} a pour élément structurant une fonction dont la forme est celle de la fenêtre utilisée dans le spectrogramme pour la fréquence 0 et qui vaut zéro ailleurs, en échelle logarithmique, *i.e.* : $\forall (t, \omega) \in \mathbb{R} \times \mathbb{R}$,

$$\sigma_4(t, \omega) = 10 \log_{10}(w(t)\chi_{\{0\}}(\omega)) \,. \qquad (28)$$

Elle sert à concentrer le bruit en temps puisqu'il s'était étalé lors de sa transformation en spectrogramme. Le résultat de cette opération est montré dans la figure 10 ; nous voyons comment l'intensité qui s'étalait autour de l'instant 1 s dans l'ouverture est concentrée après l'application de l'érosion.

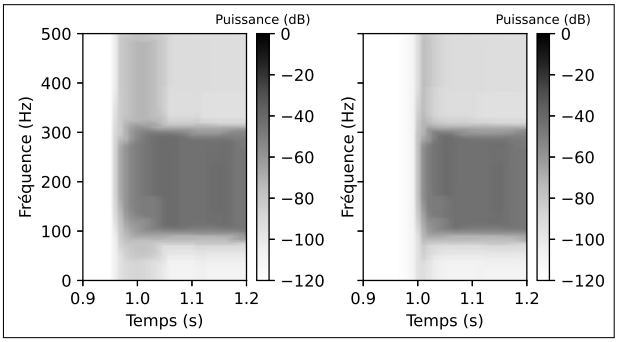


Figure 10. Ouverture (à gauche) et le résultat de l'érosion ε_{σ_4} (à droite).

5.2.4. Filtrage du bruit avec le résultat des opérations

Si nous composons les opérations morphologiques, nous obtenons un résultat $\Theta : \mathbb{R} \times \mathbb{R} \to \overline{\mathbb{R}}$ qui s'écrit

$$\Theta = \varepsilon_{\sigma_4}\left[\gamma_{\sigma_3}\left[\varphi_{\sigma_1}\left[\mathrm{SPEC}^{\mathrm{dB}}[f]\right]\right]\right] . \qquad (29)$$

Il s'agit alors de l'utiliser comme masque pour filtrer notre bruit blanc, et nous procédons comme dans [10] pour utiliser la STFT :

1. Nous générons un bruit blanc.
2. Nous le normalisons pour qu'il ait une puissance maximale [9] de 0 dB.
3. Nous calculons sa STFT.
4. Nous multiplions point par point la STFT du bruit blanc par Θ en échelle linéaire [10].
5. Nous reconstruisons le signal par STFT inverse [11].

Le résultat de ce processus nous donne un signal dont le spectrogramme est montré dans la figure 11 en le comparant à la partie non-harmonique du signal de référence. Nous retrouvons la partie non-harmonique dans notre synthèse, mais aussi du bruit supplémentaire ; celui-ci est dû à l'étalement spectral de la partie harmonique. En revanche, il se situe à une puissance aux alentours de −90 dB, donc sa contribution au signal est relativement petite.

9 . Dans la pratique, si nous normalisons pour que le maximum soit 0 dB, nous aurons une perte de puissance dans le bruit ; pour les applications, nous nous servons de la fermeture du bruit blanc pour normaliser chaque région à son maximum local.

10 . Θ est en échelle logarithmique car les spectrogrammes sont en dB ; Θ en échelle linéaire s'écrit alors $\Theta^{\mathrm{lin}} = 10^{\frac{\Theta}{10}}$.

11 . La STFT inverse nous permet de reconstruire un signal à partir de sa STFT. Pour sa définition, voir [8].

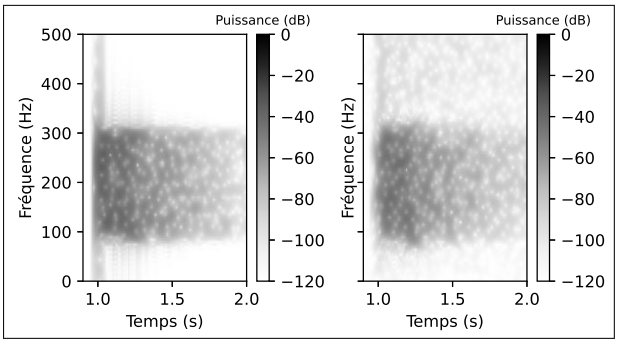


Figure 11. Partie non harmonique du signal de référence (à gauche) le bruit filtré synthétisé par notre méthode (à droite).

6. RÉSULTATS

Jusqu'à présent, nous avons testé notre méthode sur un signal de référence, obtenant un très bon résultat pour la partie harmonique et un résultat correct pour la partie non-harmonique. La méthode a ensuite été appliquée à des sons d'instruments issus la base de données TinySOL [4] qui a été récupérée grâce à la librairie `mirdata` [2].

Nous invitons le lecteur à consulter

`https://github.com/Manza12/JIM-2022.`

pour écouter les résultats. Dans le `README.md`, des instructions sont données pour l'écoute des sons et l'utilisation du code, disponible en libre accès [12].

6.1. Bois

Nous avons testé la méthode sur des notes de flûte, clarinette en si bémol, hautbois, basson et saxo alto. Les résultats sont très convaincants ; la partie harmonique est très bien resynthétisée (sauf, parfois, le transitoire d'attaque) et la partie non-harmonique simule l'effet du souffle. La figure 12 montre une comparaison des spectrogrammes des sons d'entrée et de sortie d'un La 3 de flûte.

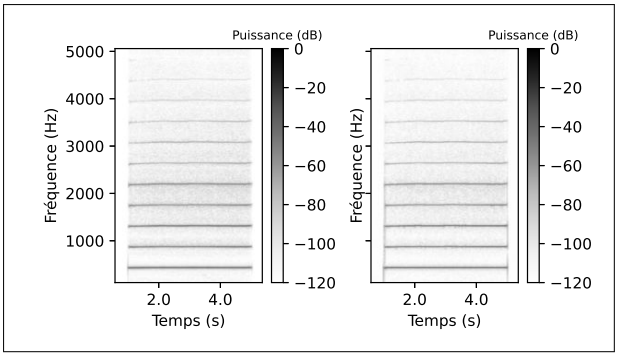


Figure 12. Comparaison du spectrogramme du signal d'entrée (à gauche) et de sortie (à droite) d'un La 3 de flûte.

12 . Licence GNU General Public License v3.0.

6.2. Cuivres

Pour les cuivres, nous avons testé la méthode sur la trompette, le cor, le trombone et le tuba. Les résultats sont moins bons, notamment en ce qui concerne le transitoire d'attaque, très caractéristique des cuivres. La partie harmonique oscille plus que dans le signal original, notamment dans les graves, à cause de la résolution fréquentielle limitée et du fait que le squelette donne des pixels qui ne sont pas toujours alignés. La partie non-harmonique correspondant au souffle est retrouvée comme dans le cas des bois. La figure 13 montre un Do 2 de trombone et le résultat de sortie.

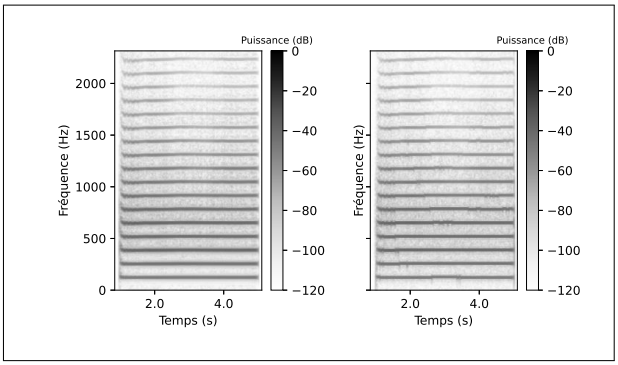


Figure 13. Comparaison du spectrogramme du signal d'entrée (à gauche) et de sortie (à droite) d'un Do 2 de trombone.

6.3. Cordes

Le cas des cordes est plus variable. Nous avons testé la méthode sur des sons de contrebasse, violoncelle, alto et violon. Les résultats dépendent de deux facteurs principaux ; le premier est la tessiture : quand les sons sont trop graves, comme dans le cas de la contrebasse, la résolution fréquentielle limitée fait que les partiels se confondent avec le bruit. Il faut alors augmenter la taille de la fenêtre de la STFT et diminuer les largeurs des éléments structurants de la fermeture et de l'ouverture pour qu'ils s'adaptent à des fréquences plus serrées. Les résultats s'améliorent mais restent moins bons que dans les tessitures plus hautes. L'autre facteur est le vibrato ; quand l'instrumentiste vibre, les résidus d'ouverture ont du mal à détecter cette modulation de fréquence rapide et rendent le squelette non connexe [13] , surtout dans les aigus. La figure 14 montre cet effet. Cela provoque des artéfacts très perceptibles. En revanche, quand le son est produit sur une corde à vide, le son est très réussi et on retrouve bien le son de l'archet dans la partie non-harmonique. Il faut noter aussi que les attaques sont bien meilleures que dans le cas des vents, puisque plus progressives.

13 . Notons qu'une squelettisation par amincissement homotopique pourrait en partie résoudre ce problème.

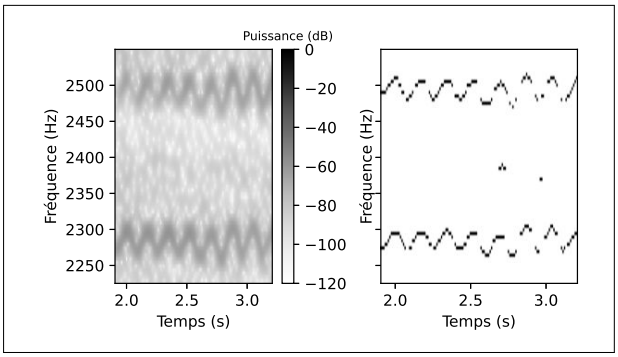


Figure 14. Comparaison du spectrogramme du signal d'entrée (à gauche) et le résultat du squelette (à droite) dans des partiels aigus d'un La bémol 2 de violon vibré.

7. CONCLUSION

Dans cet article, nous avons montré comment les opérateurs morphologiques appliqués à des spectrogrammes de sons d'instruments de musique peuvent nous aider à estimer des paramètres permettant de reconstruire les sons avec une qualité considérable, à la fois pour la partie harmonique et pour la partie non-harmonique.

Parmi ces opérateurs, le squelette des résidus d'ouverture s'avère particulièrement intéressant comme méthode pour identifier les partiels harmoniques car il parvient à réduire les lignes horizontales du spectrogramme à un pixel d'épaisseur. L'ouverture a été aussi très utile pour filtrer ces lignes et retrouver la réponse en fréquence d'un filtre qui varie dans le temps avec laquelle nous filtrons un bruit blanc.

Pour les travaux futurs, nous proposons de privilégier deux voies d'amélioration. En premier lieu, l'attaque est problématique à synthétiser par cette méthode ; une idée serait d'ajouter une composante qui lui soit particulière et voir comment estimer ses paramètres. Étant donné que l'attaque est représentée comme une ligne verticale dans un spectrogramme, nous pourrions utiliser des techniques similaires à celles de la partie harmonique, mais adaptées à la détection des lignes saillantes verticales au lieu des lignes saillantes horizontales.

En second lieu, nous pourrions améliorer la détection des partiels harmoniques en utilisant une autre technique de morphologie mathématique appelée *amincissement* [5] pour calculer un squelette homotope à l'ensemble de départ ; elle pourrait ainsi résoudre le problème de connexité des squelettes des notes vibrées. De plus, cette technique appliquée directement à l'image en niveaux de gris éviterait la binarisation de l'image.

8. REFERENCES

[1] James W. Beauchamp. Analysis and synthesis of musical instrument sounds. In *Analysis, Synthesis, and Perception of Musical Sounds*, pages 1–89. Springer, Urbana, USA, 2007.

[2] Rachel M Bittner, Magdalena Fuentes, David Rubinstein, Andreas Jansson, Keunwoo Choi, and Thor Kell. mirdata : Software for Reproducible Usage of Datasets. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*, pages 99–106, Delft, The Netherlands, 2019.

[3] Isabelle Bloch, Henk Heijmans, and Christian Ronse. Mathematical Morphology. In *Handbook of spatial logic*. Springer, Dordrecht, The Netherlands, 2007.

[4] C.E. Cella, D. Ghisi, V. Lostanlen, F. Lévy, J. Fineberg, and Y. Maresz. OrchideaSOL : a dataset of extended instrumental techniques for computer-aided orchestration. In *Proceedings of the International Computer Music Conference*, Santiago, Chile, 2020.

[5] Michel Couprie, Nivando Bezerra, and Gilles Bertrand. A Parallel Thinning Algorithm for Grayscale Images. In *Proceedings of the 17th IAPR International Conference on Discrete Geometry for Computer Imagery*, pages 71–82, Seville, Spain, 2013.

[6] Ali Daher, El Houssaïn Baghious, Gilles Burel, and Emanuel Radoi. Overlap-Save and Overlap-Add Filters : Optimal Design and Comparison. *IEEE Transactions on Signal Processing*, 58(6) :3066–3075, 2010.

[7] Jesse Engel, Lamtharn (Hanoi) Hantrakul, Chenjie Gu, and Adam Roberts. DDSP : Differentiable Digital Signal Processing. In *International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

[8] Karlheinz Gröchenig. *Foundations of Time-Frequency Analysis*. Birkhäuser, Boston, 2001.

[9] H. J. A. M Heijmans and C Ronse. The algebraic basis of mathematical morphology I. Dilations and erosions. *Computer Vision, Graphics, and Image Processing*, 50(3) :245–295, 1990.

[10] Nian-chyi Huang and J. K. Aggarwal. Time-varying digital signal processing. In *Proceedings of the 19th IEEE Conference on Decision and Control, Including the Symposium on Adaptive Processes*, pages 586–587, Albuquerque, USA, 1980.

[11] Christian Lantuejoul. *La squelettisation et son application aux mesures topologiques des mosaiques polycristallines*. PhD thesis, École des Mines, Paris, 1978.

[12] Georges Matheron. *Random sets and integral geometry*. J. Wiley & Sons, New York, USA, 1975.

[13] Lawrence R. Rabiner and Bernard Gold. *Theory and Application of Digital Signal Processing*. Prentice Hall, Englewood Cliffs, USA, 1975.

[14] Richard A. Roberts and Clifford T. Mullis. *Digital signal processing*. Addison-Wesley, Reading, USA, 1987.

[15] C. Ronse and H. J. A. M. Heijmans. The algebraic basis of mathematical morphology : II. Openings and closings. *CVGIP : Image Understanding*, 54(1) :74–97, 1991.

[16] Jean P. Serra. *Image analysis and mathematical morphology*. Academic Press, Orlando, USA, 1982.

[17] Xavier Serra. Musical Sound Modeling with Sinusoids plus Noise. In *Musical Signal Processing*, pages 91–122. Routledge, New York, USA, 1997.

[18] Xavier Serra and Julius Smith. Spectral modeling synthesis : A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4) :12–24, 1990.

SMC-22 Remote Demo Session

SOUND SCOPE PAD

Masatoshi Hamanaka
RIKEN
masatoshi.hamanaka@riken.jp

ABSTRACT

We developed Sound Scope Pad to provide an active music listening experience that combines AI, virtual reality (VR), and spatial acoustics. Users can emphasize the sounds of different performers by turning their head to the left or right or bringing their hands closer to their ears, allowing them to find and focus on the performer they want to hear. In the Sound Scope Headphones that we previously built, the user's head direction was detected by an accelerometer mounted on the headphones arc. In Sound Scope Pad, we enable detection in the head direction by combining the angle information detected by the acceleration gyro sensor of a tablet and the angle information of the head recognized from the front camera image of the tablet.

1. INTRODUCTION

Sound Scope Pad is an application that enables users to focus on the sound of a specific part of a song more clearly while listening to it (Fig. 1). Turning their head to the left or right enables users to follow the sound of specific instruments on the respective sides. Moving their palms close to their ears in a "listening pose" enables users to focus on only specific parts.

We previously proposed Sound Scope Headphones that achieved the aforementioned functions by using a digital compass that detects the orientation of the face and distance sensors that measure the distance between a hand and an ear [1, 2]. Figure 2 shows our exhibition at ACM SIGGRAPH2009 Emerging Technologies. Since face-to-face exhibitions are currently difficult due to social distancing in the wake of COVID-19, we investigated implementing similar functions in an app so that many people can experience it using a tablet.

2. RELATED WORKS

Headphones equipped with sensors that detect the direction and position of the head have been available for a while, but since their purpose is to enhance the sense of presence by fixing the virtual sound source position [3–5], they cannot be used to emphasize a specific part. For example, these headphones make it difficult to selectively listen to only a certain instrument located nearby other instruments.

Copyright: © 2022 Masatoshi Hamanaka. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any médium, provided the original author and source are credited.

Figure 1. Exhibition of Sound Scope Headphones

The spatial audio system used enables the volume of an instrument to be changed by moving the position of the listener's avatar and each part [6, 7]. However, it can be difficult for beginners to properly adjust the mixing of each part because this requires a complicated operation in which the part where a solo start must be placed in the center localization and the part where it returns to the accompaniment must be placed far away.

3. SOUND SCOPE PAD

By installing the Sound Scope Pad application on a tablet, users can experience the following (Fig. 2).

VR Concert Experience Performers stand around you and start playing, and you can see three at a time in front of you on your iPad screen. If you point the iPad to the left, the performer will appear on the left, and if you point it to the right, the performer will appear on the right.

Spatial Acoustic Experience If you hold the iPad in front of you, you will see the application tracking your face on the display. By turning your head to the left or right, you can follow the sound of the performers on their respective sides. Likewise, if you turn around with the iPad in hand, you will be able to face the performers behind you.

Active Music Listening Experience This app provides an active music listening experience by enabling you to focus on certain performers that you want to hear. If you turn to face the performer of your choice, it will sound as though they are directly in front of you. By moving your left or right palm closer or further from your ear, you can adjust the range of the performance that you want to focus on. You can highlight one part or multiple parts of the performance, such as the horn section or the rhythm section. You can check the direction of the performer and the direction you are facing on the position display.

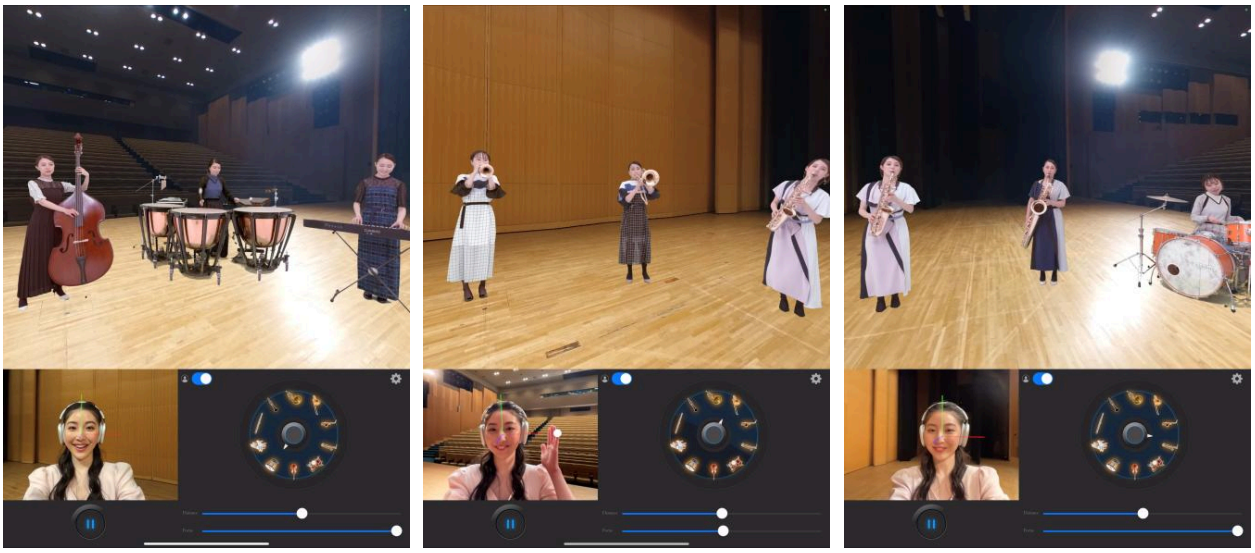


Figure 2. Screen snapshot of Sound Scope Pad.

4. IMPLEMENTATION

Let θ_n be the angle created from each of the n parts from the listener's avatar. By changing the amplification rate $h_n{}^{\theta}$ ($0 \leq h_n{}^{\theta} \leq 1$) in accordance with the orientation of the head, the sound of the part at an angle close to the front can be more emphasized. In the following, the angle between the head direction and the position where the part is placed is θ ($-\pi \leq \theta < -\pi$), and the distance between the hand and ear is δ ($0 \leq \delta \leq 1$).

$$h_n{}^{\theta} = \begin{cases} 0 & \widetilde{h_n{}^{\theta}} < 0 \\ \widetilde{h_n{}^{\theta}} & 0 \geq \widetilde{h_n{}^{\theta}} \end{cases} \quad (1),$$

where

$$h_n{}^{\theta} = \begin{cases} 0 & \widetilde{h_n{}^{\theta}} < 0 \\ \widetilde{h_n{}^{\theta}} & 0 \geq \widetilde{h_n{}^{\theta}} \end{cases} \quad (2).$$

α ($0 \leq \alpha \leq 1$) is an adjustable parameter that sets the change in amplification rate when $\delta < 1$. When $\alpha = 0$, the amplification rate of each part does not change even if the hand is closer to the ear, but when $\alpha > 0$, the amplification rate decreases as the hand approaches the ear. At this time, since the decrease in the amplification rate is larger in the direction in which the listener is not facing, the sound in front can be heard relatively loudly.

5. CONCLUSION

We have described Sound Scope Pad, an application that enables users to emphasize the part they want to listen to in a song consisting of multiple parts by using head direction and hand gestures. The Sound Scope Pad app and an introductory video explaining how to use it can be downloaded at

`https://gttm.jp/hamanaka/en/soundscopepad/`

6. REFERENCES

[1] Masatoshi Hamanaka and Sunghee Lee, "Sound Scope Headphones," ACM SIGGRAPH, 2009, Talks TK-201 / Emerging Technologies ET-201.

[2] Masatoshi Hamanaka and Sunghee Lee, "Music Scope Headphones: Natural User Interface for Selection of Music," In Proc. of the 7th Int. Conf. on Music Information Retrieval, Victoria, Canada, 2006, pp. 302–307.

[3] Oliver Warusfel and Gerhard Eckel, "LISTEN: Augmenting Everyday Environments Through Interactive Soundscapes," In Proc. of IEEE Workshop on VR for Public Consumption, IEEE Virtual Reality, 2004, Chicago, USA, pp. 268–275.

[4] Jiann-Rong Wu, Cha-Dong Duh, M. Ouhyoung, and Jei-Tun Wu, "Head Motion and Latency Compensation on Localization of 3D Sound in Virtual Reality," In Proc. of the ACM Symposium on Virtual Reality Software and Technology, ACM Virtual Reality Software and Technology, 1997, Lausanne, Switzerland, pp. 15–20.

[5] Camille Goudeseune and Hank Kaczmarski, "Composing Outdoor Augmented-Reality Sound Environments," In Proc. of the Int. Computer Music Conf., International Computer Music Association, 2001, Havana, Cuba, pp. 83–86.

[6] François Pachet and Oliver Delerue, "A Mixed 2D/3D Interface for Music Spatialization," In Lecture Notes in Computer Science (no. 1434) First Int. Conf. on Virtual Worlds, 1998, Paris, France, pp. 298–307.

[7] François Pachet and Oliver Delerue. "On-The-Fly Multi-Track Mixing," In Proceedings of AES 109th Convention , Audio Engineering Society, 2000, Los Angeles.

Distracted - A piece for Violin and an Audio-Visual Interface

Anna Savery
University of Technology Sydney
anna.savery@student.uts.edu.au

ABSTRACT

This is an extended abstract for a demo of an audio-visual interface for a violin bow demonstrated through a composition called *Distracted*.

1. INTRODUCTION

The author's primary PhD research at the University of Technology, Sydney focuses on the design of a wireless audio-visual interface, fitted to the frog of a violin bow.

The interface is designed to be a discrete tool that allows an expert practitioner to expand their creative tool bank beyond the limitations of a traditional acoustic instrument by combining their existing skill set with the ever expanding possibilities of technology. Embedded in practiced-based research, the interface is developing through composition, improvisation and live performances. Hence, the first of many compositions was written to explore the possibilities of the prototype.

This paper will focus on the first of three movements of *Distracted - Rain*. A six minute structure, starting from solo violin, then building layers through live sampling and audio processing and pre-recorded playback. The visual component, made up of video samples, is also processed and manipulated in real-time, as one, cohesive improvisation with some pre-determined flag posts. There is a predetermined structure, enabled by timed triggers, initiated by the first toggle that turns on the sound within the max patch. Some instances are enabled directly by the interface, others rely on pitch detection, or slightly more random data from the sensors. It is a fine balance between predetermined, carefully planed out, reliable moments, and surprising, unpredictable instances, allowing for improvisation and spontaneous variations.

2. BRIEF OVERVIEW OF THE AUGMENTED VIOLIN

Although the modern violin has remained essentially the same since it was standardized in the 18c (Steinberg, 2002 [1])the pursuit of implementing extended techniques and augmenting the instrument have led to an abundance of studies and innovations.

Copyright: © 2022 Anna Savery et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2.1 The Electric and the Augmented Acoustic Violin

In their book, *The Contemporary Violin: Extended Performence Technique*, Strange et al. [2] outline the history of the electric violin and amplification of the instrument, dating back to as early as 1930, with the first commercially available electric violin being made by Rickenbaker, the manufacturer of electric guitars, in 1935. The development of electric violins expanded to the addition of extra strings, creating a hybrid viola/cello violin, such as the six string The Viper, created by rock violinist Mark Wood (Lieberman, 1997) [3]

The use of commercially available pickups, microphones and amplifiers was game changing for string players, allowing them to manipulate their sound whilst retaining the feel and technique of the original acoustic instrument. The ability to amplify an acoustic instrument also allowed violinists and other quieter instruments to stand their ground and be heard within a non-traditional setting, such as a jazz or rock band, which would otherwise be impossible.

In more recent years, research has been published into gathering data from the acoustic violin for analysis and creative purposes. In 2006, Bevilacqua et al. published their research on the augmented violin developed at IRCAM which was an acoustic violin with "added sensing capabilities to measure the bow acceleration in realtime." [4] In 2013, Gidion et al., published their research on using polymer sensor technology to study the dynamic patterns of the violin bridge by mounting the sensors under the feet of the bridge. The motion of the two bridge feet was captured separately, so that their mutual phase and amplitude relation could be followed while the violin was played in the accustomed way. They were able to gain a deeper understanding of how the instrument produces sound as well as the complexities of the fluctuating intensities of harmonics depending on their location on the fingerboard. [4]

Dan Trueman and Perry R Cook's BoSSA (Bowed-Sensor-Speaker-Array), [5] is a completely novel instrument that was put together after years of previous research which broke the violin down into separate components and created a completely new looking instrument. Challenging the audience's preconceived associations with the violin.

Unlike the developments mentioned above, the author's interface is specifically designed to fill the void for an elegant, wireless solution aimed at a virtuoso string player that may also identify as improviser and composer. As demonstrated in *Distracted*, the possibility of sound manipulation in real-time is very fluid. Fitted with four pushbuttons, two slider potentiometers and an accelerometer sensor, the interface communicates wirelessly with

Max/MSP and Jitter through the Xbee/ZigBee module, using an arduino Fio. The 3D printed casing allows for a light yet sturdy design.

The Arduino code along with the Max patch came together organically as the physicality of the interface began to take shape. The initial stages consisted of simple tests of data inputs and outputs, making sure all the software and hardware was communicating properly. In it's early stages of development, the interface functioned as a very basic loop pedal, initiating and ending sample recording and playback. Then, as the work began to take shape, so did the ideas and possibilities of the prototype.

3. PHYSICALITY OF THE INTERFACE

The audio-visual interface is made up of an Arduino Fio, with an XBee radio receiver mounted on top.

There is a small rechargeable lithium battery that is tucked away between the XBee module and the Fio. The other components are a push button module with four buttons, two, small slider potentiometers and an accelleromter sensor. All the hardware lives inside a 3D printed casing that sits around the frog of the bow. The Fio radio sends data to an XBee receiver which is mounted to an XBee expolerer USB and connected to a computer. *Distracted* is written using basic Arduino code and a Max/MSP and Jitter patch. Arduino communicates when the buttons are on or off, relevant data from the sliders and the acceleromter. But all the interesting and creative aspects of the piece happen in Max.

As with any new instrument, there is a learning curve. Designing, experimenting and practicing with the interface requires time and patience, often resulting in frustration. However, the result, is a truly unique, fully customized extension of an instrument, already capable of deep expression and technical prowess. Limited only by the engineering and programming capabilities of the engineer.

In *Distracted - Rain*, there is one pre-recorded sample, spliced together in Logic and read through a buffer . All other samples are recorded in real-time and processed using the interface, as well as some mapped instances within the patch.

For now, the audio signal is sent using a pickup, mounted around the violin. This pickup has a 1/4 inch jack input and the cable is inserted into an audio interface that is connected to a computer. A future goal, would be a completely wireless set up, with no cables attached to the acoustic instrument. The justification for this goal, is a visual aesthetic of elegance, freedom and wonder. An aesthetic which a visible cable connected to a black belt like pick-up around the instrument, completely destroys.

4. MUSIC STRUCTURE OF THE WORK

The arc of this movement is simple. It starts thin, with a solo violin, and builds as more samples are layered on top and around the acoustic melody. There is a pre-composed melody, but it is lose and free. The same goes for the harmonic structure. There are worked out chord land posts, but they are reached freely and no performance is alike. The piece grows and develops organically. Apart from the rain and wind sample, everything else is fluid.

Partly influenced by the rainfall and storms in NSW during the summer of 2022, this movement explores personal challenges imposed by extreme weather conditions and uncontrollable external circumstances.

5. VISUAL TECHNICALITIES

Made up of home videos and stock footage, the visual component juxtaposes together the cognitive difficulties induced by prolonged periods of rainfall against the playfulness of children in such circumstance.

Similarly to the audio manipulation, the visuals are processed and triggered using the interface. The data from the Arduino is sent to Max, and the video component is executed in Jitter.

The visuals are displayed on a large scrim, with the performer standing and moving around behind it. Lighting is used to accentuate either the artist or the visuals, depending on their importance in the current instance within the piece, creating one cohesive audio-visual experience.

6. CONCLUSIONS

This extended abstract discussed the author's ongoing PhD research into augmenting the violin through the design and engineering of a wireless audio-visual interface, fitted onto the frog of the bow. The workings of the interface are demonstrated through a three part composition named *Distracted*, focusing on the first movement *Rain*.

Acknowledgments

This research is supported by an Australian Government Research Training Program Scholarship.

7. REFERENCES

[1] M. Steinberg, "The Contemporary Violin: Extended Performance Techniques (review)," *Notes*, vol. 59, no. 2, pp. 352–354, 2002.

[2] P. Strange and A. Strange, *The contemporary violin: Extended performance techniques*. Scarecrow Press, 2003, vol. 7.

[3] J. L. Lieberman, *Improvising Violin*. Hal Leonard, 1997. [Online]. Available: https://books.google.com/books?id=9f8KAQAAQBAJ

[4] F. Bevilacqua, N. H. Rasamimanana, E. Fléty, S. Lemouton, and F. Baschet, "The augmented violin project: research, composition and performance report," in *6th International Conference on New Interfaces for Musical Expression (NIME 06)*, Paris, France, jun 2006, pp. 402–406. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01161349

[5] P. Cook and D. Trueman, "BoSSA: The Deconstructed Violin Reconstructed," *Journal of New Music Research*, vol. 29, no. 2, pp. 121–130, 2000.

SMC-22 Demo Session 1

SPATIALIZED POLYPHONIC GRANULAR

Emmanouil Dimogerontakis
Aalborg University
edimog21@student.aau.dk

ABSTRACT

This paper aims to present a novel interface for composing and performing with granular synthesis. The described system allows handling individual spatialization for independent groups of voices and controlling various parameters of the granular synthesis in real-time. The DMI tries to create a new interaction, outside of the norms of traditional controllers, giving the user the freedom to move the groups of grains over the space and at the same time curate the sound on a fundamental level. This paper describes the sound synthesis and the interface design as well as its various functions.

1. INTRODUCTION

Granular synthesis is a relatively new composition tool, with the rise of computer power software-based synthesis became accessible to artists and composers, and GS is used extensively nowadays. Besides, electroacoustic music introduced us to new terms and ideas about how the composer faces the sound. Musical parameters such as density of sound, timbre, space, etc. became a reality for composition and expressiveness. This created the need to find new interfaces for musical expression for this type of synthesis.

2. INSTRUMENT DESCRIPTION

2.1 Polyphonic Granular synthesis in space

The concept of granulation of sound was firstly proposed by Gabor as 'acoustical quanta' and later on by Xenakis as 'grains of sounds'. In the '80s the first software for granular was made by Roads[1][2][3]. Early experimentation formed different techniques such as asynchronous GS, synchronous GS, FOG synthesis, etc. And today, there are programs and software written in various musical and programming languages. Most of them are VSTs or standalone programs and the UIs are based on knobs, sliders, XY controllers, etc. The user controls them with the mouse and the keyboard in most cases. Other interfaces for HCI can be found in GS like GR-1 or Collidoscope[4].

The presented DMI is written in the musical programming language MAX/MSP[5]. The main goal for this DMI is to deliver a qualitative granular engine with minimal and straightforward controls. Previous works for controlling the GS in space and creating grain clouds can be found here[6].

The engine behind it consists of a polyphonic grain generator. Grains are defined as small-windowed portions of sound with a duration from 10ms to 100ms. Except for the grain size, other parameters of GS are the position in the audio buffer, this can be imaged as the pointer of the audio buffer, the playback speed of the grains, the pitch, the position in the space(if we are choosing the stereo format this parameter is the panning) the amplitude of the grains and the window type. With polyphony we mean the instances of the same grain, we will refer to them as grain voices, these grain voices are forming the grain clouds. In MAX/MSP the polyphony can be created easily with the [poly~] object.

2.2 Users Interface

The main goal of this project was the creation of a new interface for polyphonic GS with direct controls both in spatialization and in the parameters of GS. With a minimal interface, the artist can expand his creativity without spending time understanding the controls and the mappings.

The instrument consists of two trackpads. With the left hand, the artist controls the position of the grains in the space and with the right hand, the artist controls the grain size, the location in the buffer, the playback speed, and the pitch.

On the left, for the spatialization of the grains, it is used the Sensel Morph[7]. With the specific trackpad, it is possible to have 3-dimensional controls with one touch, it tracks the position of the finger (in XY axis) and the pressure of the touch. Additionally, Sensel Morph detects multiple contacts and tracks the pressure sensitivity and position for each contact. In this case, each finger can control several grains in space, thus you can think of the Sensel Morph as a map with coordinates that you can arrange objects in a room.

On the right hand, another simple trackpad is used to control the mentioned parameters of the GS synthesis. Here, the trackpad is divided into 4 smaller horizontal areas and four different vectors are created. With this, fader-like controls are formed for manipulating the four parameters of the GS synthesis (Grain size, Location, Playback speed, Pitch). The difference between the created controller and faders is that the user can 'jump' from one value to another without interpolation. For example, if the system has values from 0 to 10 with the presented control the user could choose any value directly, on the other hand with a fader the system would need to read all the previous values to select the requested value. Additionally, the tangible medium to control the trackpad is a pen.

The participation in the conference was supported by the European Art Science and Technology Network (EASTN-DC).

Figure 1. The UI. On the left, the Sensel Morph is located, and on the right is the trackpad for controlling the parameters of the GS synthesis.

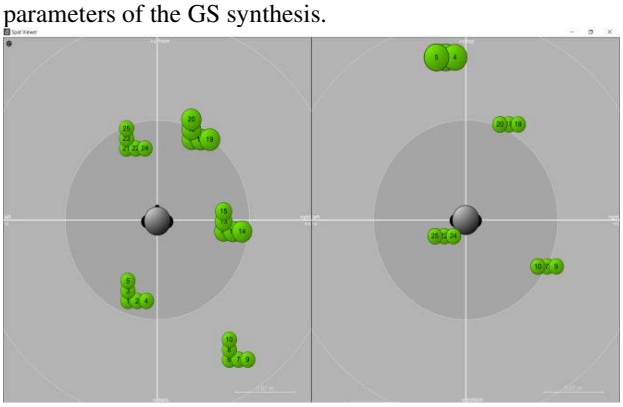


Figure 2. A screenshot of the spat.viewer. On the left side, there are the grain groups controlled by the fingers on XY axis (position in the space), and on the right the same grain groups on XZ axis (levitation).

2.3 Mappings

As we previously discussed, with the space controller (Sensel Morph) the user can control and move into the space the grains of the GS synthesis. The mappings of fingers and grains are direct. For each finger 5-grain voices are mapped, although the number could be changed. With this function, the user can create and control a grain cloud.

Depending on the format for the spatialization that is chosen the pressure sensitivity can be used as levitation control (in the case of binaural for example). The integration between the Sensel Morph and MAX MSP has been done simply by using the object [sensel]. This object gives access to all data of the Sensel Morph directly and is easy to map these data with parameters in the patch. For decoding and encoding in a spatialization format, the Spat5 library[8] has been selected. With the above library, the manipulation and the control of spatialization format are immediate and straightforward and it is a very handy tool for robust results. For mapping, the values from the second trackpad to the parameters of the GS synthesis of the [serial] object were used. Again the mappings into the parameters of GS were direct and they were treated as vectors.

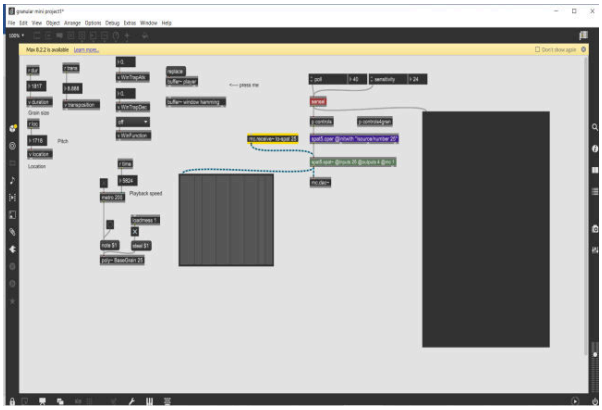


Figure 3. A screenshot of the MAX MSP patch.

3. CONCLUSIONS

This report describes a novel DMI for GS synthesis. Which gives the possibility to the user to spatialize groups of grains into space. The interface has direct controls to support the creativity of the performer/composer. A lot of new features can be added such as choosing the number of grain voices per finger, controlling the distance of each grain, and using real-time audio as an input. Moreover, improvements can be done for a more accurate design of the user's experience[9], by updating the values of GS to the controller (for example to rescale the values of the controller for different duration of the buffers), making the pitch vector logarithmic, and finding a mechanism for muting the grains when there is no contact in the Sensel Morph. And finally, the specific GS instrument could be tested in other spatialization formats to explore its expressivity.

4. REFERENCES

[1] Gabor, D. Acoustical quanta and the theory of hearing. Nature 159 (4044), 1947, 591-594.

[2] Xenakis, I. Formalized Music, Bloomington: Indiana University Press, 1971.

[3] Roads, C. Introduction to Granular Synthesis. Computer Music Journal 12, 2. Cambridge, MA, 1988.

[4] Bengler, B. Collidoscope: Let's Ride the Waves of Sound. (2016).

[5] *Cycling '74*. . https://cycling74.com/

[6] Carlson, C. & Wang, G. Borderlands: An Audiovisual Interface for Granular Synthesis. NIME '11 (University of Michigan, Ann Arbour, 2011).

[7] *Sensel Morph*. https://morph.sensel.com/

[8] Carpentier, T. (2018). *A new implementation of Spat in Max*. 184-- 191. https://hal.archives-ouvertes.fr/hal-02094499.

[9] Baine, T., Fels, S. (2003). Contexts of Collaborative Musical Experiences, A Nime Reader: Fifteen Years of New Interfaces for Musical Expression (2017).

SOCIAL / MUSICAL GAMING WITH COUNT-ME-IN

Juan Pablo Carrascal
Microsoft
jpcarrascal@acm.org

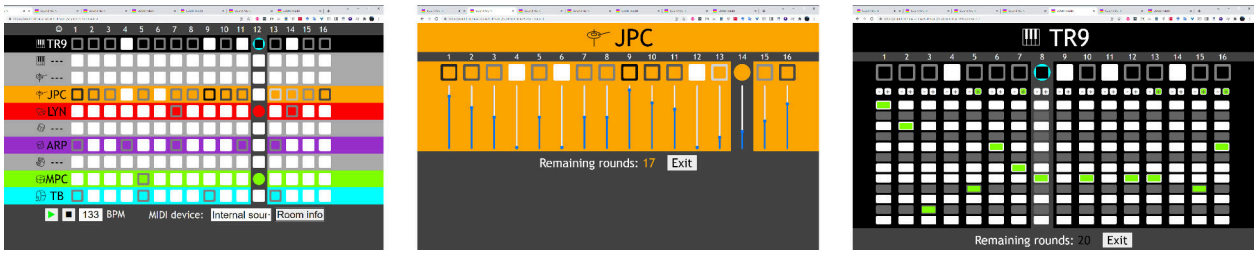


Figure 1. Count-Me-In Web apps. Left to right: the Sequencer, the Drum Track and the Synth Track.

ABSTRACT

Count-Me-In is a collaborative music sequencer allows a group of people to create music together in a playful manner. By using a Web application architecture that allows access from any Internet-connected device (including mobiles), it is very easy to take part in a Count-Me-In session. While this demo focuses on the playful use of Count-Me-In in a social context, it can also promote active audience participation in music performances, installations, and other related contexts.

1. APPLICATION DESIGN AND DEMO DESCRIPTION

Count-Me-In [1] is a collaborative networked music sequencer meant to be used in a shared space using a public display and participants' personal devices. It consists of 3 different app types: the Sequencer, the Drum Track and the Synth Track. A 10-track step Sequencer web app is displayed publicly, while participants join the session using their devices by scanning a QR code or entering a session URL. Once they join, participants are randomly assigned a Track app (either a Drum or a Synth instrument) as they enter a session. The Track apps allow participants to control the steps of a single track at a time. This way, participants can collaboratively build a music loop and hear the result in real time. In order to accommodate for a number of people larger than the number of tracks, the time that each participant can work on a track is limited. Once that time is over, the track is released and made available for other participant to use.

The Sequencer,depicted in Figure 1 (left), features ten tracks and 16 steps, and it is meant to be shown in the public display. The Drum and Synth Track applications are shown in Figure 1 (middle and right, respectively). The current prototype features eight Drum Tracks and two Synth Tracks, but this can be configured by software.

Copyright: © 2022 Juan Pablo Carrascal et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. DEMO REQUIREMENTS

A few elements are required for presenting the demo during the conference: 1) A computer (provided by the presenter), 2) a large display (or a video projector with a projection screen) with HDMI input; 3) a stable Internet connection; 4) a sound amplification system.

Attendees will use their own devices to participate in the demo, so a stable WiFi network is also recommended.

3. REFERENCES

[1] J. P. Carrascal, "Count-me-in: A collaborative step sequencer for audience participation," in *19th Sound and Music Computing Conference, SMC 2022*. Sound and Music Computing Network, 2022.

Figure 2. Playful Count-Me-In session in an informal gathering.

Playing the Virtual Glass Armonica

Astrid Pedersen, Morten Jørgensen, Halfdan Isaksen, Mikkel Riedel
Aalborg University Copenhagen
aekp19,mojarg19,hisaks19,mriede19@student.aau.dk

ABSTRACT

The continuation of research and technological advances within the field of virtual reality (VR) and haptics, further facilitates new interaction opportunities, which can be created for virtual reality musical instruments (VRMIs). Integrated in a museum setting, this can allow for new ways for both users and scholars to interact with, and learn about, musical artifacts, which otherwise would be contained behind glass cases. In this paper, the authors design and implement a virtual Glass Armonica, synthesized with the FAUST Physical Modeling ToolKit, to be played with the wearable haptic interface, WeArt TouchDiver; providing force-, temperature- and vibrotactile feedback to enhance playability. This is done with the purpose of providing an immersive cultural experience in a historically accurate virtual environment (VE), for the visitors at the Danish Music Museum in Copenhagen.

1. INTRODUCTION

The interactive and multimodal experiences enabled by VR platforms, continues to gain prominence as a medium for artistic expression. Development in the field of musical interaction design has enabled researchers to explore novel affordances of creatively expressive musical interfaces. Furthermore, as the rapid development and availability of interactive technologies reaches public establishments, VR experiences are increasingly being included in museum institutions to further assist the cultural experience [1].

In collaboration with the Danish Music Museum, this paper aims to present the theoretical reasoning, and implementation of an interactive musical interface, replicating the Glass Armonica in an immersive VE, communicating the rich history of the instrument. Playing the Glass Armonica much resembles that of rubbing the edge of a wineglass with a moist finger. The instrument, invented by Benjamin Franklin, adopts glass bowls turned by a horizontal axle. This way, the side of the bowls are submerged into a trough of water [2]. As playing the instrument produces multiple tactile cues, the haptic device of choice was the WeArt TouchDiver, which provides haptic actuators much aligning with the haptic sensations of playing the original counterpart.

Copyright: © 2022 Astrid Pedersen et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. RELATED WORKS

Musical interfaces in VR has been implemented in museums in order to both promote and increase accessibility of historical instruments, in ways not normally accessible to the visitor. In [3], two mixed reality applications were developed to promote the *Guqin*, a traditional Chinese string instrument. A field study revealed an overwhelming increase in interest of the instrument after using their VR system. Likewise in [4], traditional instruments were simulated and recreated in a culturally communicative VE to enhance the users' sense of on-site experience; which increased the depth of cultural communication. VR experiences thereby not only has the potential to display the instrument multimodally, but also its historical context, communicated through the immersive VE.

The expressive musical interfaces have been coined as VRMIs [5]. Following the definition, Serafin et al. establishes a set of nine conceptual design principles; first of which discusses the design for feedback and mapping. When playing an acoustic instrument, an intense haptic experience is not only unavoidable, but also highly relevant to the playability [5]. The correlation between auditory and haptic modalities has been a widespread topic of research, and has received increasing interest in the field of sound and music computing [6]. In [7], Avanzini et al. suggests a multimodal architecture, integrating physically-based sound models with haptic and visual rendering. Their experimental results revealed evidence of how auditory feedback can modulate tactile perception of contact stiffness. In the works of [8], Papetti et al. emphasizes the role of haptics in the complex action–perception loop for instrument performance, and furthermore highlights the absence between sound production and active haptic feedback for current digital musical instruments.

In the context of cultural heritage, providing haptic feedback would potentially not only increase the playability of the musical interface, but also enable the visitor to touch the past, possibly offering an immersive cultural experience [5, 9].

3. HAPTIC INTERFACE

For reproducing haptic feedback aligning with that of playing the Glass Armonica in real life, the haptic device WeArt TouchDiver was employed. The vibrotactile actuators provides a bandwidth for texture rendering (20-500Hz), thermal feedback enabling material discrimination with 0.1°C resolution, and up to 7N for skin indentation with resolution of 0,05N; on the tips of the thumb, index, and mid-

Figure 1. First iteration 3D model of the Glass Armonica

dle fingers. The user plays the Glass Armonica with continuous contact, as the finger must apply the appropriate amount of continuous force, before actuating vibrations and producing sound. When replicating the musical interface, the mapping between the haptic output is therefore relevant for the playability, as the force-actuated vibrations will guide the user to produce the auditory feedback. Additionally, as the instrument require application of water before being able to produce sound, this introduces the potential of including thermal feedback. How this contributes to the playability is uncertain, but if a realistic rendering of the interaction is desired, it would be conceivable to include.

4. SOUND SYNTHESIS AND IMPLEMENTATION

The sounds of the Glass Armonica were simulated using FAUST Physical Modeling ToolKit. The toolkit provides a set of virtual musical instruments based on physical modelling. The notes assigned to the playable bowls were approximated from listening to Glass Armonica samples. An overarching benefit of physical modelling, is that the physical parameters of the model can be controlled by user actions [7]. In the perspective of the Glass Armonica, the physical parameter of Bow_Pressure was mapped to the force index of the haptic device, controlled by the location of the virtual hand. Ultimately letting the user control the amplitude of the sound produced, based on pressure applied onto the virtual glass bowl.

The 3D instrument, seen on figure 1, and other models in the VE, were modelled using Blender before being implemented in Unity 3D. The WeArt TouchDiver offers an SDK compatible with Unity, facilitating easy control and mapping of the actuators featured in the device. For hand-tracking, the haptic device was mounted on an HTC Vive Tracker, and visuals were rendered through an HTC Vive Pro head-mounted display.

5. CONCLUSIONS

As the technological development of interactive experiences increases accessibility, VR is more commonly being implemented in public establishment; such as museums. Based on existing design frameworks of VRMIs, the design of feedback for the Glass Armonica using active haptics, is now increasingly conceivable with the WeArt TouchDiver.

Implementing an interactive musical interface will, in the context of cultural heritage, not only provide access for visitors to experience the acoustics of the instrument, but also the historical context; giving way for not only musical expressivity in VR, but also a platform for cultural dissemination.

Acknowledgments

The participation to the conference was supported by the European Art Science and Technology Network (EASTN-DC).

6. REFERENCES

[1] R. Gaugne, F. Nouviale, O. Rioual, A. Chirat, K. Gohon, V. Goupil, M. Toutirais, B. Bossis, and V. Gouranton, "Evoluson: Walking through an interactive history of music," *Presence*, vol. 26, no. 03, pp. 281–296, 2018.

[2] R. Paisa and S. Serafin, "Recreating an instrument inspired by the glass harmonica using fabrication techniques and physical models," in *15th Sound and Music Computing Conference, SMC 2018*. Sound and Music Computing Network, 2018, pp. 239–242.

[3] M. Yu, M. Zhang, C. Yu, X. Ma, X.-D. Yang, and J. Zhang, "We can do more to save guqin: Design and evaluate interactive systems to make guqin more accessible to the general public," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–12.

[4] L. Wu, W. Su, S. Ye, and R. Yu, "Digital museum for traditional culture showcase and interactive experience based on virtual reality," in *2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*. IEEE, 2021, pp. 218–223.

[5] S. Serafin, C. Erkut, J. Kojs, N. C. Nilsson, and R. Nordahl, "Virtual reality musical instruments: State of the art, design principles, and future directions," *Computer Music Journal*, vol. 40, no. 3, pp. 22–40, 2016.

[6] A. Passalenti, R. Paisa, N. C. Nilsson, N. S. Andersson, F. Fontana, R. Nordahl, and S. Serafin, "No strings attached: Force and vibrotactile feedback in a guitar simulation," in *Proceedings of the 16th Sound and Music Computing Conference (SMC 2019), Málaga, Spain*, 2019, pp. 28–31.

[7] F. Avanzini and P. Crosato, "Integrating physically based sound models in a multimodal rendering architecture," *Computer Animation and Virtual Worlds*, vol. 17, no. 3-4, pp. 411–419, 2006.

[8] S. Papetti and C. Saitis, *Musical haptics*. Springer Nature, 2018.

[9] T. Nicolas, R. Gaugne, C. Tavernier, Q. Petit, V. Gouranton, and B. Arnaldi, "Touching and interacting with inaccessible cultural heritage," *Presence*, vol. 24, no. 3, pp. 265–277, 2015.

AIM-package : a modular standard for structuring patches in Max

Vincent Goudard
vincent@vincentgoudard.com

ABSTRACT

Designing a complete in-car audio experience requires solutions for rapid prototyping in a complex audio configuration, bringing together different areas of expertise ranging from sound-design and composition, down to hardware protection, with every conceivable layer of audio-engineering in-between, up to A-B comparisons setups for end-users perception evaluation in real demonstration vehicules. The AIM-package started as a request from the "Active Sound Experience" team at Volvo Cars Company [1] to meet such goals.

To this end, it was decided to develop a framework on top of Max [2] so that dedicated audio processing modules could be easily created, with the ability to store presets for various configurations, and to take advantage of Max's modular design to distribute the complexity of audio engineering among the various expert teams involved in the project.

The AIM-package was much designed after two older Max packages, namely *Jamoma Modular* — a part of the Jamoma project [1] started in 2006, and its current continuation in *libossia*, a part of the OSSIA project [2] [3] . Their main goal is well described in [1] : "to address concerns of sharing and exchanging Max patches in a modular system. This means creating a structured framework that does enforce consistency, readability, and standards for interoperability while not placing daunting restrictions on users." Beyond this goal, these two projects have a lot in common, with libossia being actively developed by a number of people previously involved in Jamoma. They both adopt a MVC design [4] , along with a number of conventions that contribute to their integration in the Max eco-system.

While the AIM-package is following a lot of these conventions, it was however decided to develop an alternative package —rather than just using libossia, for a number of reasons, that will be discussed here below and during the demo. Since the features of Jamoma have been described in [1], I will only stress a few differences and let the reader refer to the mentioned reference for the similarities.

[1] `https://www.volvocars.com`
[2] `https://cycling74.com`
[3] `http://www.jamoma.org` and `https://ossia.io`
[4] Model View Controler

Copyright: © 2022 Vincent Goudard et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Explicit model nesting
Contrary to Jamoma and libossia, models in AIM are explicitly bound to a parent model. This allows for a more flexible patcher organization, more consistent with the MVC design, and the possibility to dynamically rebind a model to another parent.

Typed modules
AIM modules are typed. This allows to share presets between all instances of modules of the same type. Namely, any preset created in a given model instance will readily be available to any other instance of that model, even if nested in another component.

Multichannel, multiplicity
The AIM-package is oriented towards multichannel processing [5] . To this end, it adopts a number of existing conventions to address MC objects in Max, and also supports wildcard and brace-expansion notation [6] to easily create or address multiplicities of nodes in the namespace tree.

Vanilla Max only
The AIM-package only relies on vanilla-Max objects. This generally proves to be more sustainable, as there is not C-coded external object, which compilation might break after a system update. The main drawback of course, is a worse performance and loading time, but these were acceptable enough for making this choice.

Acknowledgments

The development of the AIM-package was fully supported by Volvo Cars Company, which consider to release it as a free and open-source package for the community. For more information on the ASX project, please contact Jonatan Ewald [7] at Volvo Cars.

1. REFERENCES

[1] T. Place and T. Lossius, "Jamoma: A modular standard for structuring patches in max," in *ICMC*, 2006.

[2] J.-M. Celerier, "Authoring interactive media: a logical & temporal approach," Ph.D. dissertation, Bordeaux, 2018.

[5] ...as enabled with the introduction of MC.* objects in Max.
[6] `https://www.gnu.org/software/bash/manual/html_node/Brace-Expansion.html`
[7] jonatan.ewald@volvocars.com

THE DEVELOPMENT OF A REAL-TIME MOVEMENT SONIFICATION EXERGAME FOR BODY-WEIGHT SQUAT TRAINING

Cherelle Connor
Sound and Music Computing
Aalborg University, Copenhagen
cco@create.aau.dk

Prithvi Ravi Kantan
Sound and Music Computing
Aalborg University, Copenhagen
prka@create.aau.dk

Stefania Serafin
Sound and Music Computing
Aalborg University, Copenhagen
sts@create.aau.dk

ABSTRACT

Participating in physical activities is often difficult for individuals living with blindness or other visual impairments. The use of exergames has shown promise in affording these individuals engaging and novel methods for participation in physical activity. Specifically, movement sonification is one method shown to be reliable in providing guidance and helping users orient their bodies in space. Through this research we aim to develop an auditory-only exergame that provides augmented feedback, using a combination of verbal instruction and real-time movement sonification, for low to no vision users to learn to perform body-weight squat movements correctly and safely. We anticipate that this research will assist in further establishing the importance of movement sonification feedback for better exercise training and comprehension in physical activity when no visual input is present.

1. INTRODUCTION

Individuals living with blindness or visual impairments often report having difficulty participating in physical activity due to fear of injury and inexperience. [1] They have limited options for exercise instruction, many of which are resource-intensive and costly. [2] As a result, these individuals often report being significantly less active than sighted individuals, which can lead to an increase in obesity and an overall deterioration in health. [1] Recent advancements in exergame development have begun to address these concerns, creating applications that are accessible to the visually impaired via auditory interfaces. [3]

2. BACKGROUND & RELATED WORK

2.1 Exergames & Visual Impairment

Exergames are virtual games designed to encourage physical activity and fitness through entertainment. [4] These games have also become a valuable tool in motor skill development. [2] Not only have they been shown to be entertaining, but they also lower the barrier for learning and engagement in new activities. [2, 4] This is especially important in developing accessible exergames, as it allows users to feel comfort and confidence as they use an application. [3] However, there is still a lack of exergames designed for individuals with visual impairments. [4] This is likely due to the difficulty in overcoming the challenges of designing an interactive application without the use of a visual interface. [3] A 2017 study by Rector et. al., introduced Eyes-Free Yoga, a yoga exergame designed for blind users, that provided step-by-step verbal guidance on the performance of yoga poses. [2] In their paper, Rector reported difficulty in determining what verbal descriptors would be most informative for body positioning and instruction. Considering that some users were not previously familiar with the selected yoga poses, they experienced difficulty in translating the verbal instructions into body orientations and movements in space. The addition of non-verbal auditory feedback information, specifically movement sonification, can address this gap as it has been show to enhance a user's understanding of movement descriptors [5] and temporal and spatial information [6], which can result in improved movement techniques and overall performance. [7]

2.2 Movement Sonification

Movement sonification refers to the transformation of motion signals into sound. [8] Users can listen to generated sounds mapped to bodily motions, allowing for a greater understanding of how and where their body is moving in space, and providing an effective technique for improvement of physical performance. [9] This approach has proven especially beneficial for rehabilitation, dance and sports technique improvement. [10] Still, there is limited research on how movement sonification can be used to provide instruction for accessible exercise learning and not solely improvement. Integrating real-time movement sonification, in addition to standard verbal instruction, into an exergame can create an alternative method of exercise learning and training for individuals with visual impairments, offering them richer feedback for skill acquisition. [10] Therefore, we aim to develop a system that provides real-time movement sonification feedback for exercise instruction that ensures the user is performing the exercise safely and correctly.

Copyright: © 2022 Cherelle Connor et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

3. DESIGN METHODS

3.1 Exercise Selection & Safety

For the purposes of this study, we chose to develop our system around the body-weight squat exercise. We chose this movement as it is fundamental in many activities of daily living such as, sitting and lifting and because it serves as a basis for more dynamic tasks involved in other physical activities. [7] This application is designed for no to low vision users with normal hearing. We conferred with a physiotherapist and personal trainer to ensure the appropriate feedback is being presented. To provide better instruction, we have compartmentalized the squat movement according to four main criteria gathered from our expert interviews: foot placement, knee alignment, knee flexion, and weight shifting. The system provides real-time movement sonification feedback on the performance of the aforementioned criteria.

3.2 Technical Development

The application's auditory interface is being developed in Unity3D Game Engine and the desired sounds are generated using Faust programming language. Navigation of the application occurs via voice commands to allow for interaction without a visual interface. The necessary body tracking information, for bodily motion mapping, is extracted using the Azure Kinect's Body Tracking SDK.

Within the application, verbal instruction and movement sonification provide a step-by-step guide on how to properly execute a body-weight squat. Once the user understands the fundamentals of the movement, the system monitors their performance of the squat and returns sonic feedback for real-time adjustment of the performance, if needed. The system will then assist the user to focus on each area of improvement individually and sonify each of these areas independently to avoid overly dense feedback. For the squat exercise, the areas of improvement revolved around the ankles, knees, hips, and back, as well as the shifting of weight throughout the movement.The angular changes and weight shifting are mapped to sonic renderings using the Faust Unity plugin.

4. EXPECTED CONTRIBUTIONS

We anticipate the main contribution of this work will be the creation of an auditory-only exergame that emphasizes real-time movement sonification feedback for body-weight squat learning and performance that can be operated without the need for in-person training. The system will be designed to ensure the safety of its users. We believe this system can improve the user's motivation and confidence in their ability to pursue physical exercise and better health. Lastly, in future work, we plan conduct a user study to evaluate the system. We also plan to expand this workflow to additional exercises, in order to create a full, thorough workout regimen.

Acknowledgments

I would like to thank Dr. Stefania Serafin and Prithvi Ravi Kantan who have graciously hosted and guided me during my time at the Multisensory Experience Lab. My participation in this conference was supported by the American Scandanavian Fellowship Committee and the European Art Science and Technology Network (EASTN-DC).

5. REFERENCES

[1] M. Capella-McDonnall, "The need for health promotion for adults who are visually impaired," *Journal of Visual Impairment & Blindness*, vol. 101, no. 3, pp. 133–145, 2007.

[2] K. Rector, R. Vilardaga, L. Lansky, K. Lu, C. L. Bennett, R. E. Ladner, and J. A. Kientz, "Design and real-world evaluation of eyes-free yoga: an exergame for blind and low-vision exercise," *ACM Transactions on Accessible Computing (TACCESS)*, vol. 9, no. 4, pp. 1–25, 2017.

[3] G. Giannakopoulos, N.-A. Tatlas, V. Giannakopoulos, A. Floros, and P. Katsoulis, "Accessible electronic games for blind children and young people," *British Journal of Educational Technology*, vol. 49, no. 4, pp. 608–619, 2018.

[4] N. I. Othman, N. A. M. Zin, and H. Mohamed, "Accessibility requirements in serious games for low vision children," in *2019 International Conference on Electrical Engineering and Informatics (ICEEI)*. IEEE, 2019, pp. 624–630.

[5] F. Bevilacqua, E. O. Boyer, J. Françoise, O. Houix, P. Susini, A. Roby-Brami, and S. Hanneton, "Sensorimotor learning with movement sonification: perspectives from recent interdisciplinary studies," *Frontiers in neuroscience*, vol. 10, p. 385, 2016.

[6] G. Rosati, F. Oscari, S. Spagnol, F. Avanzini, and S. Masiero, "Effect of task-related continuous auditory feedback during learning of tracking motion exercises," *Journal of neuroengineering and rehabilitation*, vol. 9, no. 1, pp. 1–13, 2012.

[7] R. Hale, *Validation of an error sonification auditory feedback training program on proper sagittal plane squat technique*. The University of Texas at El Paso, 2016.

[8] J. F. Dyer, "Human movement sonification for motor skill learning," Ph.D. dissertation, Queen's University Belfast. Faculty of Engineering and Physical Sciences, 2017.

[9] A. O. Effenberg, U. Fehse, G. Schmitz, B. Krueger, and H. Mechling, "Movement sonification: effects on motor learning beyond rhythmic adjustments," *Frontiers in neuroscience*, vol. 10, p. 219, 2016.

[10] N. Schaffert, T. B. Janzen, K. Mattes, and M. H. Thaut, "A review on the relationship between sound and movement in sports and rehabilitation," *Frontiers in psychology*, vol. 10, p. 244, 2019.

A Virtual Reality Volumetric Music Video: Featuring New Pagans

Gareth W. Young
V-SENSE
YoungGa@tcd.ie

Néill O'Dwyer
V-SENSE
ODwyerNC@tcd.ie

Aljosa Smolic
V-SENSE
SmolicA@tcd.ie

ABSTRACT

Music videos are short films that integrate songs and imagery produced for artistic and promotional purposes. Modern music videos apply various media capture techniques and creative post-production technologies to provide a myriad of stimulating and artistic approaches to audience entertainment and engagement for viewing across multiple devices. Within this domain, volumetric video (VV) capture technologies (Figure 1) have become an emerging means of recording and reproducing musical performances for new audiences to access via traditional 2D screens and emergent extended reality (XR) platforms, such as augmented and virtual reality (AR/VR). These 3D digital reproductions of musical performances are captured live and are enhanced to deliver cutting-edge audiovisual entertainment. However, the precise impact of VV in music video entertainment is still in a state of flux.

1. SUMMARY

The proposed VR Volumetric Music Video demonstration will help establish how users respond to VV representations of music performance via VR technology. As a sophisticated, interactive music video that can be accessed and presented via multiple XR platforms, we will demonstrate new workflows on how volumetric music videos may be captured, edited, and accessed for virtual live performance. This approach to contemporary music interactions will show how audiences are likely to react to music videos in an XR context and offer insights into how future music video research may be further developed.

2. ARTIST STATEMENT

Finding new ways to visualize and communicate musical performance in VR is driven by artistic creativity, a desire to innovate technologically, and a need to capture new and existing audience attention. It has long been accepted that "Artificial reality is the authentic postmodern condition, and virtual reality its definitive technological expression" [1, p.169]; therefore, it stands to reason that postmodernist art representations can be expressed within VR. Immersive volumetric music videos are being studied at V-SENSE as an emergent art form in and of themselves

Copyright: © 2022 Gareth W. Young et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Figure 1. Volumetric Video Capture

as many emergent XR technologies are being applied in this endeavor, such as stereoscopic and 360° audiovisual spatial recording technology. These capture technologies have expanded the traditional viewing medium to include further dimensions of immersion, interaction, and imagination for the audience and were closely tied to advancements in home PC GPU/CPU speeds, HMD optics, software data processing capabilities, and AI.

Figure 2. Scenes from within the proposed VR demonstration.

The study of XR music videos has been used to inform V-SENSE's user-centered design of a custom-made VV VR music video experience, featuring the New Pagans' track Lily Yeats (Figure 2). The project's pilot study initially highlighted the specific qualities that audiences seek during the consumption of such materials [2]. Iterations of this novel application area are expected to focus on differences between traditional media and new XR experiences and expose and build upon existing HCI studies that focus on music and technology in use, specifically those concerning how users experience music videos presented via 6DoF XR technologies.

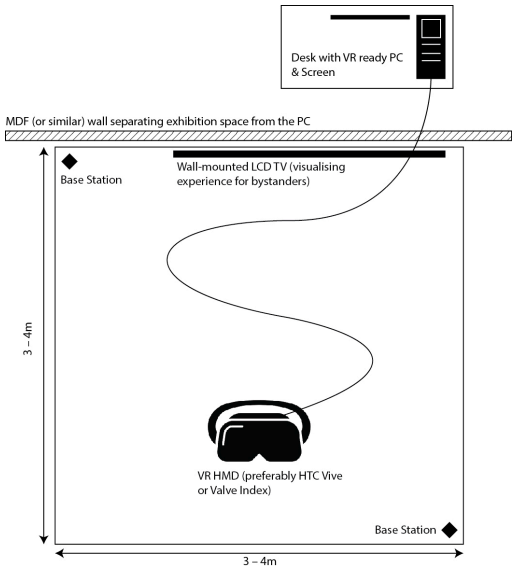


Figure 3. Floor plan

3. TECHNICAL DESCRIPTION

The authors propose demonstrating the outputs from our ongoing research in this area to the delegates of SMC 2022 that can attend in person. Therefore, a virtual reality installation for the conference is presented that uses (dynamic) VV and (static) 3D world-building techniques combined and displayed using the Unity game engine. Figure 3 highlights the specific requirements for viewing.

4. ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of the Science Foundation Ireland (SFI) under Grant Number 15/RP/2776.

5. REFERENCES

[1] B. Woolley, *Virtual worlds: A journey in hype and hyperreality*. London: Penguin, 1993.

[2] G. W. Young, N. O'Dwyer, and A. Smolic, "Audience experiences of a volumetric virtual reality music video," in *2021 IEEE Conference on Virtual Reality and 3D User Interfaces*. Christchurch: IEEE, 2022, pp. 775 – 781.

MusCIcal Adventures: Virtual Reality as a Musical Training Program

Emil Sønderskov Hansen
Aalborg University CPH
esha19@student.aau.dk

Signe Marie Kromann Kristiansen
Aalborg University CPH
smkk19@student.aau.dk

Ali Adjorlu
Multisensory Experience Lab
adj@create.aau.dk

Stefania Serafin
Multisensory Experience Lab
sts@create.aau.dk

ABSTRACT

Training programs for improving abilities in speech for CI users are vastly common. However, musical training programs (MTP) for improving music perception is still limited, resulting in CI users having difficulties perceiving, and thereby enjoying, music. Many CI users describe listening to music as an unfulfilling experience, especially when it comes to recognizing melodies and identifying various musical instruments, among other aspects. Additionally, determining the direction of sound is another difficulty CI users often tend to have. Although MTPs have been made utilizing tablet based games, DVD-formats etc., using virtual reality environments for training various aspects of music recognition has not yet been explored. This applies for using VR for spatial sound recognition as well. In this paper, a VR prototype functioning as a MTP with gamification elements will be introduced, with the purpose of improving various musical perception abilities for CI users.

1. INTRODUCTION

Cochlear implants (CI) are advanced systems used for individuals with complete or severe hearing loss to regain functional hearing by stimulating the auditory nerve with electric current [1]. CI users can expect to recover recognition of speech and the ability to differentiate sounds, however, the degree of recovery differs and requires training [2]. Since current rehabilitation training post implantation mainly focuses on training speech and awareness of environmental sounds [3], music perception has not yet been explored to the same degree. Therefore, focusing on improving the listening abilities in regards to both musical perception and sound localization for CI users could be beneficial in gaining a more pleasant music listening experience.

Copyright: © 2022 Emil Sønderskov Hansen, Signe Marie Kromann Kristiansen et. al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. MUSICAL PERCEPTION AND MUSIC TRAINING PROGRAMS FOR CI USERS

Many studies have shown CI users can recognize rhythm on a comparable level as normal hearing (NH) individuals [4] [5]. However, when it comes to identifying melodies, CI users generally tend to have more difficulties compared to NH listeners [5].

Previous research in music training has included timbre recognition [6], identification of musical instruments [7], pitch perception [8] and melodic contour identification (MCI) [5], where CI users abilities in these musical perception aspects improved over time. However, [9] argues the pleasantness of listening to music only increased after focused training for a longer period of time [9].

3. PITCH, TIMBRE AND SPATIAL LOCALIZATION WITH CI

CI users have difficulties perceiving some of the primary elements of music, including pitch and timbre [10], which is essential when identifying musical instruments and melodies. For the majority of CI users the perception of pitch is disrupted caused by the implant itself being out-of-tune [10]. As a result of this disrupt perception of pitch, CI users are only able to correctly identify approximately 19% of melodies [4]. Timbre perception in regards to recognizing musical instruments seems to be notably more difficult for CI users compared to NH individuals [11]. However, more percussive instruments with a faster attack appear to be easier recognizable for CI users [11]. Evidence points to increased music enjoyment when the processing of timbre becomes less demanding [10], indicating training to be able to increase enjoyabilty.
Unilateral CI users report having difficulties with localisation of sounds [12], since it requires perceiving differences in arrival time, frequency and intensity [13]. Research suggests NH individuals to perform 3 times better than bileteral CI users in sound localisation tasks when in VR [13]. Although bilateral CI users have less difficulties with localisation of sounds compared to unilateral users [12], they still have issues with fusing sounds from both ears [14]. Research indicates sound localisation can be improved through training with audio-visual tasks [14]

4. PROTOTYPE

MTPs for CI users have been made through tablet based games [15], DVD-format [16] and web-site based programs [7] among others, however, training musical perception in VR with head mounted displays (HMDs) appears to be a rather unexplored field of research. Therefore, this study will introduce a VR application for training musical instrument identification, melody recognition and sound localization. The VR environment will be implemented using Unity [1]. The program will utilize gamification elements such as auditory and visual affirmation feedback and scoreboards to let the user follow their progress during gameplay. Audio tracks are implemented using FMOD Studio with Steam Audio integration, incorporating Head Related Transfer Function (HRTF) for rendering 3D positional audio [2]. This allows having spatial audio in the environment, meaning the direction and source of music can be perceived. The game itself will focus on a) identification of musical instruments, b) sound localization, c) melody recognition, and has been developed as a part of the authors' bachelor's thesis with Stefania Serafin and Ali Adjorlu as supervisors.

Figure 1. Instrument identification in the prototype

5. ACKNOWLEDGEMENTS

The participation to the conference was supported by the European Art Science and Technology Network (EASTN-DC).

6. REFERENCES

[1] F.-G. Zeng, S. J. Rebscher, W. V. Harrison, X. Sun, and H. Feng, "Cochlear implants: System design, integration, and evaluation," *IEEE Reviews in Biomedical Engineering*, vol. 1, pp. 115–142, 2008.

[2] L. Friesen, R. Shannon, D. Başkent, and X. Wang, "Speech recognition in noise as a function of the number of spectral channels: Comparison of acoustic hearing and cochlear implants," *The Journal of the Acoustical Society of America*, vol. 110, pp. 1150–63, 09 2001.

[3] M. S, D. SA, L. P, H. T, and H. M, "Appreciation of music in adult patients with cochlear implants: a patient questionnaire," 2003.

[4] H. J. McDermott, "Music perception with cochlear implants: A review," *Trends in Amplification*, vol. 8, no. 2, pp. 49–82, 2004.

[1] https://unity.com/

[2] https://valvesoftware.github.io/steam-audio/doc/fmod/index.html

[5] J. J. Glavin, Q.-J. Fu, and R. V. Shannon, "Melodic contour identification and music perception by cochlear implant users," 2009.

[6] K. Gfeller, S. Witt, M. A. Mehr, G. Woodworth, and J. Knutson, "Effects of frequency, instrumental family, and cochlear implant type on timbre recognition and appraisal," *Annals of Otology, Rhinology & Laryngology*, vol. 111, no. 4, pp. 349–356, 2002.

[7] V. D. Driscoll, "The effects of training on recognition of musical instruments by adults with cochlear implants," 2012.

[8] K. E. Gfeller, G. G. Woodworth, D. A. Robin, S. A. Witt, and J. F. Knutson, "Perception of rhythmic and sequential pitch patterns by normally hearing adults and adult cochlear implant users," *Ear and Hearing*, vol. 18, p. 252–260, 1997.

[9] K. Gfeller, S. Witt, G. C. Woodworth, and K. D., "The effects of familiarity and complexity on music perception of cochlear implant recipients," 1996.

[10] N. T. Jiam, M. T. Caldwell, and C. J. Limb, "What does music sound like for a cochlear implant user?" *Otology amp; Neurotology*, vol. 38, no. 8, 2017.

[11] S. J. Kim, Y. S. Cho, E. Y. Kim, and G. E. Yoo, "Can young adolescents with cochlear implants perceive different timbral cues?" *Cochlear Implants International*, vol. 16, no. 2, p. 61–68, 2014.

[12] R. Y. Litovsky, M. J. Goupell, S. Godar, T. Grieco-Calub, G. L. Jones, S. N. Garadat, S. Agrawal, A. Kan, A. Todd, C. Hess, and et al., "Studies on bilateral cochlear implants at the university of wisconsin's binaural hearing and speech laboratory," *Journal of the American Academy of Audiology*, vol. 23, no. 06, p. 476–494, 2012.

[13] S. D. Sechler, A. L. Valdes, S. M. Waechter, C. Simoes-Franklin, L. Vianti, and R. B. Reilly, "Virtual reality sound localization testing in cochlear implant users," 2017.

[14] D. Vickers, M. Salorio-Corbetto, S. Driver, C. Rocca, Y. Levtov, K. Sum, B. Parmar, G. Dritsakis, J. Albanell Flores, D. Jiang, and et al., "Involving children and teenagers with bilateral cochlear implants in the design of the bears (both ears) virtual reality training suite improves personalization," *Frontiers in Digital Health*, vol. 3, 2021.

[15] T. Lasickas, J. S. Andersen, S. Serafin, and M. Vatti, "Cochlea: Gamifying ear training for cochlear implant users," 2021.

[16] V. Looi and J. She, "Music perception of cochlear implant users: A questionnaire, and its implications for a music training program," *International Journal of Audiology*, 2010.

Somax 2, a Reactive Multi-Agent Environment for Co-Improvisation

DEMO and PERFORMANCE PROPOSAL FOR SMC 2022

Joakim Borg — **Ircam-STMS** — Joakim.Borg@ircam.fr
Mikhail Malt — **Ircam-STMS** — Mikhail.Malt@ircam.fr
Gérard Assayag — **Ircam-STMS** — Gerard.Assayag@ircam.fr
Benny Sluchin — **Ircam** — benny.sluchin@ircam.fr
Simone Conforti — **Ircam - PAC** — simone.conforti@ircam.fr

Key Words

Key words: Machine learning, Co-improvisation, Audio Features, Co-creativity, Artificial Intelligence, Interaction Paradigms, Multi-agent systems

1. DEMO PROPOSAL

Somax 2 [1] is a multi-agent interactive system performing live machine co-improvisation with musicians, based on machine-listening, machine-learning, and generative units. The actual version [6, 7, 8] is a recent development and algorithms improvement from the former Somax version and previous work in RepMus team [3,4, 5, 9, 11, 12].
Agents provide stylistically coherent improvisations based on learned musical knowledge while continuously listening to and adapting to input from musicians or other agents in real time. The system is trained on any musical materials chosen by the user, effectively constructing a generative model (called a corpus), from which it draws its musical knowledge and improvisation skills. Corpora, inputs and outputs can be MIDI as well as audio, and inputs can be live or streamed from Midi or audio files. Somax 2 is one of the improvisation systems descending from the well-known Omax software, presented here in a totally new implementation. As such it shares with its siblings, the general loop [listen/learn/model/generate], using some form of statistical modeling that ends up in creating a highly organized memory structure from which it can navigate into new musical organizations, while keeping style coherence, rather than generating unheard sounds as other ML systems do. However Somax 2 adds a totally new versatility by being incredibly reactive to the musician decisions, and by putting its creative agents to communicate and work together in the same way, thanks to cognitively inspired interaction strategies and finely optimized concurrent architecture that make all its units smoothly cooperate together.
Somax 2 allows detailed parametric controls of its players and can even be played alone as an instrument in its own right, or even used in composition workflow. It is possible to listen to multiple sources and to create entire ensembles of agents where the user can control in detail how these agents interconnect and "influence" on each other.
Somax 2 is conceived to be a co-creative partner in the improvisational process, where the system after some minimal tuning is able to behave in a self-sufficient manner and participate to a diversity of improvisation set-ups and even installations.
This presentation will introduce the software environment, demonstrate its learning and interaction modes, explain the basic and advanced controls in the user interface, and allowing people to see and play with it.

2. PERFORMANCE PROPOSAL

To complete this demo session, we would like to kindly propose 4 short performance slots (4 min each), a musical performance (in concert), to present Somax 2 in real Artistic Situation.
The first two slots will be played by the flutist Simone Conforti with an EWI (electronic wind instrument) and the two other slots by the renowned international trombone player, Benny Sluchin. The main idea is to play twice the same situation in order to allow the audience to enjoy the changes and the similitudes that can occur in each improvisation context with Somax 2.

This action is part of the project REACH supported by the European Research Council under Horizon 2020 programme (Grant ERC-2019-ADG #883313) and project MERCI supported by Agence nationale de la Recherche (Grant ANR-19-CE33-0010)

Copyright: © 2022 First author et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any médium, provided the original author and source are credited.

3. REFERENCES

[1] Somax 2 General Information, demos, installation, internal reports: https://www.stms-lab.fr/projects/pages/somax2

[2] Gerard Assayag, George Bloch, Marc Chemillier, Arshia Cont, Shlomo Dubnov. *Omax brothers: a dynamic topology of agents for improvisation learning.* In: *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*. Santa Barbara, United States.2006. p. 125–132.

[3] Gérard Assayag. *Human-Machine Co-Creativity.* in Bernard Lubat, Gérard Assayag, Marc Chemillier. Artisticiel / Cyber-Improvisations. Phonofaune, 2021, Dialogiques d'Uzeste, 2021

[4] Laurent Bonnasse-Gahot. *Prototype de logiciel d'harmonisation/arrangement à la volée:* Somax v0, (2012).

[5] Laurent Bonnasse-Gahot. *An update on the SOMax project. Ircam-STMS*. Internal Report ANR Project Sample Orchestrator. 2014. Available online: http://repmus.ircam.fr/_media/dyci2/somax_project_lbg_2014. pdf (accessed on 22 October 2021), 2014.

[6] Joakim Borg. *Dynamic Classification Models for Human-Machine Improvisation and Composition*. Master's report, Ircam and Aalborg University Copenhagen, 2020.

[7] Joakim Borg. *The Somax 2 Software Architecture, internal technical report.* Ircam, April 2021a

[8] Joakim Borg. *The Somax 2 Theoretical Model, internal technical report.* Ircam, April 2021b

[9] Tristan Carsault. *Automatic chord extraction and musical structure prediction through semi-supervised learning, application to human-computer improvisation.* Diss. Ircam UMR STMS 9912, 2017.

[10] Tristan Carsault, Jérôme Nika, and Philippe Esling. *Using musical relationships between chord labels in automatic chord extraction tasks. arXiv preprint arXiv:1911. 04973* (2019).

[11] Tristan Carsault. *Introduction of musical knowledge and qualitative analysis in chord extraction and prediction tasks with machine learning.* Diss. Pierre and Marie Curie University, Paris, France, 2020.

[12] Tristan Carsault, Jérôme Nika, Philippe Esling, Gérard Assayag. *Combining Real-Time Extraction and Prediction of Musical Chord Progressions for Creative Applications*. Electronics, MDPI, 2021, 10 (21), pp.2634.

[13] Benjamin Feldman. *Improving the latent harmonic space of Somax with Variational Autoencoders*. Master's Thesis Centrale Supelec & STMS IRCAM. 2021. http://repmus.ircam.fr/merci/publications

[14] Jérôme Nika, Ken Déguernel, Axel Chemla—Romeu-Santos, Emmanuel Vincent, Gérard Assayag. *DYCI2 agents: merging the "free", "reactive", and "scenario-based" music generation paradigms*. International Computer Music Conference, Oct 2017, Shangai, China.

PD ON AN IOT-CLASS PROCESSOR

Miller Puckette
UCSD
msp@ucsd.edu

Kerry L. Hagan
UNiversity of Limerick
Kerry.Hagan@ul.ie

ABSTRACT

Audio installations and other sound art projects sometimes call for large numbers of inexpensive digital audio systems. We have adapted the Pure Data environment to an embedded system using the Espressif ESP32 processor. We find that it is possible to run Pd on an ESP32-based audio development board costing about 20 USD. We were able to build a complete system for a particular art project that can respond to wireless commands and synthesize a continuous stream of sound, while only dissipating about 925 milliwatts. This approach should be adaptable to a range of other possible sound art projects.

1. INTRODUCTION

This paper is intended as an update to a long-running story, in which digital audio signal processing becomes perpetually less expensive and more physically portable. In the last decade or so, things have progressed to the point that the most difficult aspect of audio processing is sometimes bringing electric power and communications to the physical location where audio is to be recorded, processed, or played. For example, in a recently built portable multichannel listening space [1], dozens of standalone receivers were placed at regularly spaced points in the inside of a dome. Rather than run power and signal cables to all the points, the more practical option was to use solar-charged batteries and WIFI connections using Audio Over Osc (AOO [2].

In such situations each node must be equipped with an inexpensive processor, powerful enough to do whatever processing is needed onsite, able to support audio I/O, and dissipating as little power as possible. One also has to either choose or write a framework for interactive control of the device over a wireless connection such as WIFI or Bluetooth.

The artist must then decide to either use an existing real-time program to provide the desired DSP and control capabilities, or else to develop a program from scratch. Here there is a trade-off between memory efficiency and design convenience. At one extreme, writing your own program from scratch is likely to be time-consuming (particularly since the debugging tools available might not be as well oiled as on a workstation)—but at the same time you get very tight control of CPU and memory usage. As we move along the spectrum of computer music environments, we encounter first csound, which is certainly the most compact and memory-efficient one in wide use today, through RTCMIX and Pure Data, then onto higher-level, more highly automated systems such as supercollider.

The impetus for the work described here was a recent sound installation project that required outputting audio from about 20 ugly throw pillows. To achieve spatial effects such as panning between pillows, their audio outputs must all be synchronized to within a millisecond or so but need not be synchronized at the sample level. Some mode of intercommunication is needed to coordinate them. Although we have not placed sensors inside the pillows, this remains an open possibility for future versions of the piece.

We chose to use Pure Data, as probably the highest-level tool that would fit into 1/2 megabyte of data memory. It has proved to be a tight but manageable fit.

Pure Data, being open-source, has been adapted to a variety of situations, some of which are ultra-portable implementations that take advantage of its reasonably small memory footprint [3]. An early example was Pure Data Anywhere ("PDA") [4], in which a version of Pd ran on portable electronic devices that typically had 8 or 16 MB of memory to share between code, data memory, and a filesystem. Here the main limitation was not memory size but floating-point speed; since the personal devices of that era did not normally contain floating-point units, the signal processing code in Pd was adapted to run on fixed-point audio samples. Such an approach would be of interest even today if one wanted to run Pd directly inside an earbud whose processors also normally offer only fixed-point computation. (These devices can run on a fraction of the power that we will be using in the work described here.) At least one fully realized project used PDA running on a Gumstick computer [5].

At about the same time a quite different project, also based on Pd, was installed in the New York Times lobby [6]. Here the 560 processors are wired (so no batteries nor wireless networking needed) but on the other hand, the installation, part of a building lobby, was designed to run for at least 50 years. There is visual as well as audible output. The cost of the project was much less constrained, but the maintainabilty requirement drove many design choices. Again, a fixed-point version of Pd was used and the audio processing was kept simple to run on the available CPU.

Over the ensuing years Pd has often been used on cellphones and tablet computers, for example using `MobMuPlat` [7]. Here the affordance is not so much the cheapness and portability of the system (typically in the

Copyright: © 2022 Miller Puckette et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

hundreds of US dollars), but the fact that many or most people using this platform will run it on their own tablet or cellphone. The artist thereby cedes a great deal of control over the listening environment and other aspects of the work, but the incremental cost and effort required to install such a system can be kept quite small.

A new option, the one we will study here, is to run Pd on a recently announced processor family, the Expressif ESP32. This comes in uniprocessor and two-processor models running between 160 and 240 MHz, with excellent floating-point performance. It is explicitly designed with embedded systems in mind, and so has memory-mapped flash memory (typically 2 or 4 megabytes), from which it runs instructions directly. Although it has only up to 520 kilobytes of RAM, this is only used as data memory and so the overall memory size might be considered as comparable to the original prototype ISPW. At first glance, then, running Pd on an ESP32-based embedded system would seem a viable option.

2. THE HARDWARE

The Espressif ESP32 system was first announced in 2016 and is now available in a wide variety of packages. Some of these packages incorporate the ESP32-VWROOM module that combines the ESP32 processor with a radio subsystem that supports WIFI and/or Bluetooth. The ESP32-VWROOM module is clocked at 160 mHz, has two processors with floating-point units, and can run an operating system, FreeRTOS.

On top of FreeRTOS, Espressif supplies a development environment and additional libraries that implement WIFI and Bluetooth, drivers for I2S, I2C, and UARTS, and the FAT filesystem that can be mounted from an SD card. (There are other goodies as well but these are all that will concern us here).

An artist wishing to incorporate the ESP32-VWROOM might wish to start with a development board with integrated audio, microSD reader, and battery management. These are available from Espressif under the names LyraT and LyraT Mini, and offer up to 2 channels of audio input and output for about 20 USD.

Alternatively, or perhaps in a later, more ambitious project, an artist can add one or more of their own audio I/O devices to a stripped-down development board such as the ESP32-VWROOM-E32, which adds little more than a USB programming port and a large number of breakout pins.

3. EXAMPLE USING THE LYRAT BOARD

The particular project that led us to port Pd to the ESP32 required assembling dozens of copies of a physically portable system having wireless communication with a nearby computer, a local filesystem, one channel of audio output, and the ability to run for 8 hours at a time on a single battery charge. Our hardware consists of a 3.7 volt, 2000 mAH rechargeable battery, a LyraT Mini board (a one-channel, reduced-footprint version of the LyraT), and a tiny loudspeaker. To conserve power we used Bluetooth instead of WIFI and wrote a small routine to send and receive Pd messages in ASCII. With Bluetooth, the microSD card, and the audio output all in use, the system dissipates about 250 milliamps of power (so about 925 milliwatts). Depending on how the battery and speaker are sourced, each system costs about 30 to 40 USD to build.

The systems are easily concealed inside our ugly throw pillows. We use a single Raspberry Pi computer to coordinate the ensemble. This central CPU is hidden within Bluetooth range of the pillows but need not operate on battery power (and so the 5 to 10 watts typically dissipated by Pis is not problematic). The Pis send control messages to each pillow over a Bluetooth virtual serial line. Among other things, messages to the pillows can contain the patch they will run, so that the patch itself can be edited and modified offline and updated on the Pis as needed. Sending the patch over Bluetooth as a sequence of ordinary Pd messages saves us having to upload it onto the many individual microSD cards and allows for easy on-site customization during the installation process.

4. CONCLUSION

It is feasible to run Pure Data on inexpensive, portable, low-power hardware systems. The user develops a Pd Patch on a general-purpose computer and subsequently runs it on the embedded system without the use of a graphical user interface. The techniques described here can feasibly be used in a wide variety of artistic applications.

5. REFERENCES

[1] W. Ritsch, "Der dom als virtuelle klangwelt," 2011. [Online]. Available: https://algo.mur.at/projects/klangdom/shor.pdf/@@download/file/shortdesc.pdf

[2] ——, "Towards a message based audio system," 2014. [Online]. Available: http://lac.linuxaudio.org/2014/papers/36.pdf

[3] H.-C. Steiner, "Building your own instrument with pd," in *Proceedings of the 1st International Pd Convention, Graz, Austria. Citeseer*, 2005.

[4] G. Geiger, "The search for usability and flexibility in computer music systems," in *Bang: Pure Data*. Hofheim am Taunus: Wolke Verlag, 2004, pp. 121–134.

[5] J. Stevens, M. Flynn, T. Humphrey *et al.*, "Distributed wireless duplex audio networks in a travelling art work: the megaphone project," 2007.

[6] M. Hansen, B. Rubin, E. Studio, H.-C. Steiner, T. Walker, and P. Electricks, "Words to look at, words to listen to: Designing a "proliphonic" display for the lobby of the new york times building," in *2008 Linux Audio Conference*. Citeseer, 2008.

[7] D. Iglesia and I. Intermedia, "The mobility is the message: The development and uses of mobmuplat," in *Pure Data Conference (PdCon16). New York*, 2016.

SMC-22 Demo Session 2

TICKLE TUNER - HAPTIC SMARTPHONE COVER FOR COCHLEAR IMPLANT USERS' MUSICAL TRAINING

Francesco Ganis
Multisensory Experience Lab
fganis20@student.aau.dk

Stefania Serafin
Multisensory Experience Lab
sts@create.aau.dk

Marianna Vatti
Oticon Medical
mvat@oticonmedical.com

ABSTRACT

Cochlear implants (CIs) allow hearing impaired individuals to understand speech with remarkable efficiency. On the other hand, they poorly perform in music perception. It may be possible to improve the music experience with the use of other senses such as touch. We present Tickle Tuner, a haptic feedback device suitable for musical training of CI users. The prototype is composed of two high-quality haptic actuators and an external Digital to Analogue Converter (DAC) hosted in a 3D printed enclosure coupled with a smartphone. We describe the design and implementation of the prototype, the analysis of its characteristics and we introduce a test bench for the design of different mappings between sound and vibrations.

1. DEMO DESCRIPTION

CIs are neuroprosthesis that allow people with severe or profound hearing loss to restore their sound perception. They are especially successful in reestablishing speech comprehension [1]. Music is a highly complex signal and, during the electrical stimulation of the auditory pathway, there are many factors that can influence CI users' access and fruition [2–4].In this demo we propose the Tickle Tuner, a vibrotactile feedback device that provides haptic information during musical training that can be performed using *ad hoc* mobile games. These applications are developed for improving CIs skills to better recognize musical features and thus increase music appreciation. The name of our prototype derives from the Tickle Talker, the first device that uses haptic feedback to aid speech understanding [5].

The smartphone cover has been 3D printed and is mainly composed by two parts: shell and handles. The shell features an adjustable rail system and a frame that holds in position the smartphone, hosts part of the cables and connects the two handles. These are the main contact area with the user's hands and are shaped in order to have a stable and comfortable grip. We initially shaped a handle's model using clay and then, thanks to photogrammetry, we obtained a 3D reconstruction. The Tickle Tuner features two HapCoil-One (Haptuator Mark II-D) actuators produced by Actronika [1] that can reproduce frequencies from 10 to 10000 Hz. They are both connected to a 3W stereo class-D amplifier PAM8403 chip on a DFR0119 board. The smartphone feeds the Haptuators through a digital to analog converter chip (DAC) that receives the audio stream through a USB-C plug. We slightly modified the DAC soldering two cables from the pin-out of the USB-C connector to retrieve DC (+5V) and ground (GND) for powering the amplifier. In this way, with a single plug the Tickle Tuner is able to retrieve the audio signal and the power recalling the *plug and play* concept.

In order to understand the acoustical characteristics of the Tickle Tuner, we measured its frequency response following the guidelines of Farina at al. [6]. We performed the measurements in an anechoic room using an analog accelerometer (Sparkfun ADXL335) on seven different areas of the device and recorded the vibrations from one single output axe of the accelerometer. The areas taken into account are: top, bottom and side of each handle and the center of the smartphone screen.

The impulse responses retrieved from the different locations are fairly similar and present some common characteristics such as a peak in the low range around 70 Hz and lower energy in the highest section of the spectra. The device provides robust low frequency vibrations and we expect that it will contribute to improve the perception of the lowest portion of the sound spectra in CIs users. In order to mitigate the unwanted resonances of our device, we designed two different filter banks: the first one is composed by 50 biquad filters to obtain 25 EQ points of the fourth order, The second one is an approximation of the first one formed of 7 biquad filters of the second order.

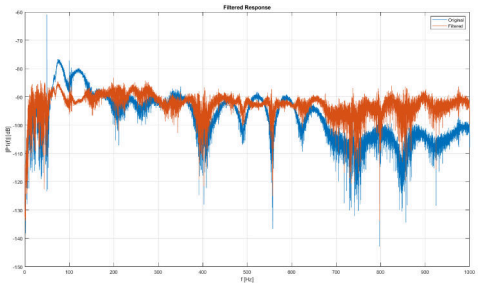


Figure 1. Comparison between the response of the Tickle Tuner with and without the 25 points EQ filtering.

Finally, we developed a third filter that compensates the

[1] https://www.actronika.com/, last access April 21, 2022

Copyright: © 2022 Francesco Ganis et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

fingertips' sensibility attenuating the mid-low range of the spectra with a center frequency of 250 Hz [7].

In order to be able to implement different mappings of haptic stimulation, we designed a prototyping environment that generates different haptic cues. The program is still under development and features generation engines and real-time feature extraction from audio signals such as pitch and envelope of the first four partials. The test bench is capable of generating and reading sounds from the smartphone to quickly prototype directly with the Tickle Tuner. We chose to use a visual programming language called Pure Data [2] since is optimal for rapid prototyping and visual interaction. We also designed a Graphical User Interface (GUI) using the platform MobMuPlat [3] to have a comfortable access to the parameters from the touchscreen. Different mappings will be coupled to different types of sounds and will be tested on both normal hearing and CIs users during the oncoming part of the project in order to assess how haptics conveyed through this prototype can help in perceiving and creating a better understanding of musical features.

Figure 2. Front and back views of the Tickle Tuner.

Acknowledgments

This work is supported by the European Art Science and Technology Network (EASTN-DC).

2. REFERENCES

[1] Fan-Gang Zeng, S. Rebscher, W. Harrison, Xiaoan Sun, and Haihong Feng, "Cochlear Implants: System Design, Integration, and Evaluation," *IEEE Reviews in Biomedical Engineering*, vol. 1, pp. 115–142, 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4664429/

[2] V. Looi, H. Mcdermott, C. M. McKay, and L. Hickson, "The effect of cochlear implantation on music perception by adults with usable pre-operative acoustic hearing," *International Journal of Audiology*, vol. 47, pp. 257 – 268, 2008.

[3] W. R. Drennan, J. J. Oleson, K. Gfeller, J. Crosson, V. D. Driscoll, J. H. Won, E. S. Anderson, and J. T. Rubinstein, "Clinical evaluation of music perception, appraisal and experience in cochlear implant users," *International Journal of Audiology*, vol. 54, no. 2, pp. 114–123, Feb. 2015. [Online]. Available: http://www.tandfonline.com/doi/full/10.3109/14992027.2014.948219

[4] C. J. Limb and A. T. Roy, "Technological, biological, and acoustical constraints to music perception in cochlear implant users," *Hearing Research*, vol. 308, pp. 13–26, Feb. 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378595513001020

[5] R. S. Cowan, J. Z. Sarant, K. L. Galvin, J. I. Alcantara, P. J. Blamey, and G. M. Clark, "The Tickle Talker: a speech perception aid for profoundly hearing impaired children," *Scientific publications, vol. 5, 1989-1990, no. 300*, 1990.

[6] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," p. 24, 2000.

[7] L. A. Jones and N. B. Sarter, "Tactile Displays: Guidance for Their Design and Application," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 50, no. 1, pp. 90–111, Feb. 2008. [Online]. Available: http://journals.sagepub.com/doi/10.1518/001872008X250638

[2] `https://puredata.info/`, last access April 21, 2022

[3] `https://danieliglesia.com/mobmuplat/`, last access April 21, 2022

Scramble Live: Combining LSTM and Markov Chains for Real-time Musical Interaction.

Nicola Privato
Conservatorio C.Pollini of Padua
nicola.privato@gmail.com

Omar Rampado
omar@ognibit.it

Alberto Novello
Conservatorio C.Pollini of Padua
alberto.novello@conservatoriopollini.it

ABSTRACT

Scramble Live is a complementary tool for *Scramble*[1], a standalone MIDI software developed in Max/MSP[2] for the real-time generation of polyphonic music [1]. At their core, both tools combine the flexibility of Markov Chains for the generation of pitch transitions with the generalization capabilities offered by LSTM neural networks on the rhythmic and dynamic domains. *Scramble* performs the analysis of MIDI files, allows for the selection of the LSTM hyperparameters and the training of the neural network through an intuitive user interface, but offers limited real-time interactivity on playback. *Scramble Live* relies on the models generated by *Scramble*, but allows for a high degree of interaction with live instrumentalists and with the system itself. To complete its performance-focused capabilities, *Scramble Live* offers MIDI mapping and synchronization to external clock sources.

INTRODUCTION

Markov Chains (MC) and Recurrent Neural Networks (RNNs) are two of the most common approaches to algorithmic composition by the means of computer-based applications [2]. Each of the two has specific advantages and limitations: RNNs can be very consistent in their predictions and perform well on generalization tasks. On the other hand they require long training time and large datasets, and do not offer much flexibility once they are trained. MCs provide a greater degree of real-time control over the produced output [3] but are inherently incapable of generalizing [4]. Additionally, the more musical dimensions are incorporated in each state (e.g. pitch and velocity), the less likely transitions become.

Scramble and *Scramble Live* take advantage of the specificities of both approaches. By combining a MC for the generation of pitch transitions with a *Long Short-Term Memory* (LSTM) RNN [5] for all time-related and dynamic variables, the real-time interaction is preserved, and the neural network's training time is reduced. Furthermore, it is possible to include in the system the live input from any MIDI instrument while offering satisfactory generalization capabilities and a high degree of rhythmic and expressive variability depending on the selected LSTM model.

We chose to develop both *Scramble*'s iterations in Max/MSP because, even though the platform offers some machine learning libraries [6], the application of more complex algorithms such as LSTM RNNs remains for the most part unexplored.

1. SCRAMBLE LIVE

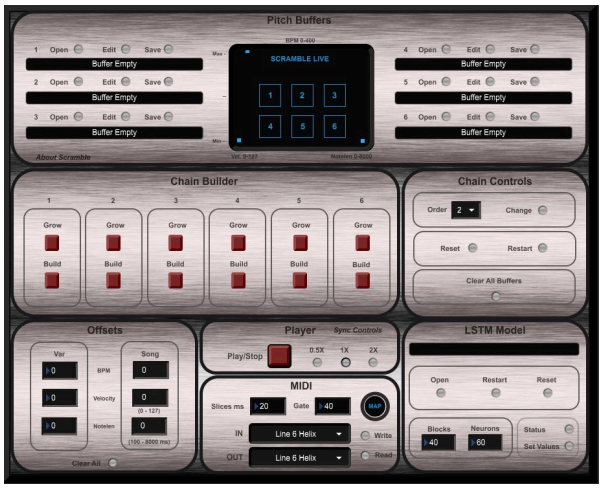


Figure 1. *Scramble Live*'s GUI.

Scramble Live is a complementary tool for *Scramble*. It is focused on live use, offering the musician an intuitive *Graphical User Interface* (GUI) that can be easily navigated while performing. In order to make the user's experience as intuitive as possible, we did not include into *Scramble Live*'s GUI the MIDI player and LSTM training sections. All LSTM models providing rhythm, tempo and velocity data to the pitch transitions generated by the MCs, must be therefore trained on and stored from *Scramble*, and subsequently opened in *Scramble Live.*

The tool offers six buffers where it is possible to temporarily store, save and load the pitch information of any MIDI sequence analyzed and saved from the main software's MIDI player, or played live from an instrument connected to the MIDI input. The data of any number of buffers, in any desired order, can be manually edited and used to generate a MC transition table. It is

[1] https://www.dropbox.com/sh/v869gify6pm6ta7/AAAcEjlWj-kIJ2VpyNfCWHN7a?dl=0
[2] https://cycling74.com/products/max

possible, for instance, to combine the pitches of a Bach invention with the real-time input of a pianist, coherently paired by the LSTM neural network with the rhythm, tempo and velocity of a model trained on Scott Joplin's ragtime style.

Furthermore, *Scramble Live* offers the possibility to change the MC transition table in real time without compromising the rhythmic structure, thus allowing the user to continuously interact with the tool and/or with any musician connected to the MIDI input.

Common to both iterations, and particularly effective in changing the MIDI output to the user's need, is the *Offsets* menu, which allows to add a positive or negative offset to the BPM, note lengths and velocity values generated by the LSTM, or to change playback to half or double time. In addition, *Scramble Live* offers the possibility to bypass the LSTM BPM variations and connect to an external master clock, in order to easily integrate the tool with any given electroacoustic setup.

In order to allow for the use of *Scramble Live* through hardware MIDI controllers, an automatic mapping function has been added. The pairing is performed by activating the mapping, selecting a playback, MC or buffer control on the GUI, and pressing the desired hardware switch. The mapping configuration can then be stored and opened at will.

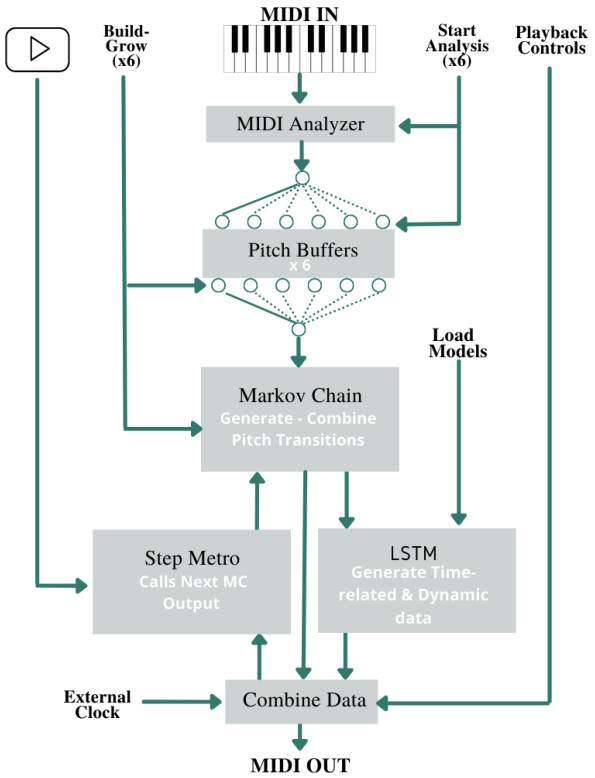


Figure 2. *Scramble Live*'s block diagram.

2. FUTURE DEVELOPMENTS

We presented *Scramble Live*, a tool for the real-time dynamic generation of polyphonic music, that extends the capabilities of *Scramble* to the performative scope. The system is based on two Max/MSP external modules: an obsolete LSTM developed by Wesley Jackson (2011)[3] and a MC from the *ml.star* library by Benjamin Day Smith [7]. The LSTM module initially presented some major bugs and design limitations that have been fixed or worked around, such as compatibility with Max 8, import and export methods, separate threading implementation and introduction of batch load for all training data. We are currently working on a new LSTM external that would allow for higher performances and architectural flexibility. The new LSTM iteration will be integrated in *Scramble* and *Scramble Live*, and become the core of a Max/MSP RNN package.

1. REFERENCES

[1] N. Privato, O. Rampado, and A. Novello, "A creative tool for the musician combining LSTM and Markov Chains in Max/MSP" in Proc. Int. Conf. on Artificial intelligence in Music, Sound, Art and Design (EvoMUSART2022), Madrid, 2022, (to be published after the conference -April 22nd 2022).

[2] H. Kumar and B. Ravindran, Polyphonic music composition with lstm neural networks and reinforcement learning. arXiv preprint arXiv: 1902.01973, 2019.

[3] A. Cleermans, D. Servan-Schreiber, and J.L. McClelland, Finite State automata and Simple Recurrent Networks. Neural Computation 1(3), 372-381(091989). https://doi.org/10.1162/neco.1989.1.3.372

[4] A. Papadopoulos, P. Roy, and F. Pachet, "Avoiding plagiarism in markov sequence generation" in Proceedings of the AAAI Conference on Artificial Intelligence. vol.28, 2022.

[5] S. Hochreiter, J. Schmidhuber, and A.S. McGough, Long-short term memory. Neural computation 9(8), 1735-1780, 1997.

[6] I. Simon, A. Roberts, C. Raffael, J. Engel, C. Hawthorne and D. Eck, Learning a latent space of multitrack measures. arXiv preprint arXiv: 1806.00195v1, 2018.

[7] B.D Smith and G.E. Garnett, "Unsupervised Play: Machine learning toolkit for max". NIME 2012.

[3] From J. Franklin's source code (2004), based on F. Gers, N. Schraudolph and J. Schmidhuber pseudocode (2002)

WAVETABLE-INSPIRED ARTIFICIAL NEURAL NETWORK SYNTHESIS

Nicholas Solem

University of California, San Diego
nsolem@ucsd.edu

ABSTRACT

We present WTIANNS, a method of database-driven sound synthesis with significant resemblance to wavetable synthesis. The primary difference between the technique presented and standard wavetable synthesis is that our method uses a multilayer perceptron (MLP) to generate wavetables in realtime. The MLP provides a continuous mapping from a set of psychoacoustic timbral features of an analyzed single-cycle waveform segment to the harmonic amplitudes of that waveform segment. In particular, we use either Bark spectrum or Bark Frequency Cepstral Coefficients (BFCCs) as input parameters. The results of the mapping are highly dependent on the character and timbral variance of the analyzed sound.

Two significant advantages of our method over wavetable synthesis include its accommodation of higher dimensionality and the fact that the waveforms are automatically derived from a pre-existing recording. WTIANNS is not intended to *emulate* the recording it is based on, but to generate waveforms that are related–whether by interpolation or extrapolation–to those of the recording. This method encourages sound designers to explore the timbre space of recordings in a manner that is divorced from time.

1. WAVETABLE-INSPIRED ARTIFICIAL NEURAL NETWORK SYNTHESIS

The method described here, denoted WTIANNS[1], involves three phases: (1) offline timbral analysis, (2) offline ANN training, and (3) realtime sound synthesis. In order to perform an accurate analysis for WTIANNS, we assume that the analyzed sound is periodic and monophonic.

1.1 Offline Timbral Analysis

The first phase is a conversion of a time-series waveform into metadata-tagged frames that describe multidimensional timbral characteristics, plus pitch, inharmonicity, and loudness. One may ask why pitch and loudness should be extracted, since the aim is to create a continuous *timbre* space, and timbre is separate from these features. The reason is that timbres of varying pitch or loudness are not directly comparable; timbres can only be compared given a constant pitch and loudness.

Rather than use a pitch detection algorithm for this work, we used recordings that are already tagged with pitch. The file names include the pitch class and octave, which is then converted to a fundamental frequency f_0 in hertz via string mapping. We do, however, detect approximate loudness via Stevens' power law [1].

The timbral features used to control the synthesized waveforms currently include the Bark Spectrum and the Bark-Frequency Cepstral Coefficients (BFCCs). [2]. The Bark spectrum provides a psychoacoustically-distributed description of spectral energy. The BFCCs express the patterns in a Bark spectrum via discrete cosine transform (DCT). The first BFCC describes overall energy, so it is not useful for timbre description; thus we discard it. The next BFCC estimates brightness, as it shows the weighting of the spectrum towards lower or higher frequencies. The following BFCC is a measurement of extremity in the low and high frequencies relative to mid-range frequencies. The BFCCs become less intuitive beyond this, but follow a pattern of fitting varying wavelength cosines to the Bark-distributed spectrum. We experiment with several truncations of the Barks/BFCCs as input parameters for resynthesis, using 3, 5, or 20 coefficients.

After tagging the frames with these features, each frame is resampled so that one wavelength is exactly represented within the frame. This step is crucial to making WTIANNS 'wavetable-inspired': in wavetable synthesis, each waveform is generally given the same number of samples, and the current time sample is an interpolation function of a phase increment dependent on frequency.

Each of the resampled waveforms is transformed into the frequency domain via Fast Fourier Transform (FFT). The DC component, along with the bins above Nyquist are discarded. We keep the magnitude spectrum and discard phase. This ensures that crossfading between waveforms causes no phase cancellation, and simplifies the training process if phase can be ignored.

The f_0, loudness, inharmonicity, and Barks/BFCCs will be used as inputs to a multilayer perceptron, and the modified spectrum will be used as the outputs. Before training the network, we perform several preprocessing steps.

For the first step, a ceiling is placed on the high harmonics for prevention of noise in training and of aliasing in resynthesis. During the resampling of each single-cycle waveform, the interpolation scheme may add artifacts–partials exceeding the frequency at which Nyquist was remapped to. These upper harmonics, though low in am-

[1] Code and samples of WTIANNS available at https://nvssynthesis.github.io/wtianns.html

plitude, would decrease the accuracy of training. Also, if the observation is resynthesized at it original frequency, these additional partials would exceed Nyquist. Thus, we null out bins above this frequency.

Next, any observations that fall outside of a dynamic range are discarded from the set. This range is based on an optional high and low percentile of the loudnesses. This ensures that we do not train the ANN on noisy data considered to be silence or any loud 'microphone bumps' that may have corrupted the observation set. For the work presented, we omit observations with loudness less than the 10th percentile of the set.

The following step is based on the fact that we hear amplitudes on a nonlinear scale. Thus, the ANN should ideally have a matching error measure. For this, we raise each bin to the power of 0.3, which approximates sones as long as the amplitude exceeds 40 dB SPL [2].

Another step taken is a truncation of the output spectrum. Regardless of the analysis window size, once these frames have been resampled, the majority of the bins beyond some high frequency should be close to zero. It will be inefficient to train the neural network on this information, and likewise inefficient to calculate all of these outputs in realtime synthesis. Thus, we truncate the spectra to a number of bins on the order of 40. This may be adjusted based on the source material analyzed.

The final preprocessing step is an output normalization. A neural network will be able to learn more consistently and accurately if the outputs occupy the whole range of the associated activation function. Therefore, we calculate the minimum and range of each bin through the set of observations, and normalize each bin as follows:

$$D_{norm} = \frac{D - \min(D)}{\max(D) - \min(D)} \tag{1}$$

This only affects the training of the ANN without affecting the resynthesis, because during synthesis, the outputs will be denormalized as

$$\hat{D} = D_{norm} \cdot (\max(D) - \min(D)) + \min(D) \tag{2}$$

1.2 Offline ANN Training

This metadata file is then used to train the ANN with the C library, FANN, which implements a multilayer feedforward ANN [3]. For the present work, we have settled on a 4-layer network (input layer, 2 hidden layers, and output layer). Depending on the truncation of the BFCCs/Bark spectrum, the input layer possesses 6, 8, or 23 neurons, corresponding to the number of coefficients, plus pitch, inharmonicity, and loudness. The output layer will have 40 neurons, corresponding with the number of harmonics we represent.

The optimal number of neurons for the two hidden layers is not yet known, but a general structure of data compression followed by expansion has yielded good results. We have experimented with many arrangements of neurons for hidden layers, but the specific number of neurons chosen seems to be of little consequence.

At present, this work makes use of only the sigmoid activation function for each neuron, despite current consensus that there are superior activation functions such as ReLU and leaky ReLU. This is simply due to the fact that the FANN library does not contain these functions.

1.3 Realtime Sound Synthesis

The WTIANNS sound synthesis itself is fairly straightforward. A voice of the synthesizer is responsible for producing a periodic phasor at the desired f_0, and for calculating waveforms any time the performer requests a new timbre. These requests happen any time the control parameters change. The control parameters correspond with the inputs of the neural network: pitch, inharmonicity, loudness, and BFCCs/Barks.

The phasor is used to simultaneously read from two lookup tables containing wavetables. Over the course of an audio block, a linear crossfade is performed to transition from one wavetable to another.

New waveforms are calculated as follows. When the user requests a new timbre, this list is sent as an input to the ANN, which computes an associated set of outputs (bin amplitudes). These outputs are raised to the power of $3.\overline{33}$, to reverse the prior spectral shaping in preprocessing (the power of 0.3). Then, the outputs are denormalized as in equation (2). These outputs are copied into a frequency lookup table, starting from the first index after DC. This frequency table should have a total length that is a power of 2 so that it can be transformed using an FFT. One more step is necessary before the FFT: the current spectrum must be reflected around the Nyquist component in order to represent the negative frequencies. There is no need to take a complex conjugate because phase has been discarded. Next, the FFT is taken from the frequency table. This renders a time domain single-cycle waveform, which is interpolated with cubic interpolation.

In the future, we hope to explore the use of Keras instead of FANN and test if better accuracy can be attained with non-sigmoidal activation functions. We also hope to test more psychoacoustic parameters. Finally, we would like to expand our method to include non-harmonic components, and use synthesis methods other than wavetable.

Copyright: © 2022 Nicholas Solem. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any médium, provided the original author and source are credited.

2. REFERENCES

[1] H. Fastl and E. Zwicker, "Loudness," in Psychoacoustics: Facts and Models, H. Fastl and E. Zwicker, Eds. Berlin, Heidelberg: Springer, 2007, pp. 203–238.

[2] E. Zwicker, G. Flottorp, and S. S. Stevens, "Critical band width in loudness summation," The Journal of the Acoustical Society of America, vol. 29, no. 5, pp. 548–557, May 1957.

[3] Nissen, S., "Implementation of a fast artificial neural network library (fann)," Report, Department of Computer Science University of Copenhagen (DIKU), 31(29), 26, 2003.

CREATING EXPERIMENTS WITH COSMONOTE: ADVANCING WEB-BASED ANNOTATIONS FOR PERFORMED MUSIC

Daniel Bedoya, Lawrence Fyfe, Elaine Chew
STMS Laboratoire (UMR9912) – CNRS, IRCAM, Sorbonne Université, Ministère de la Culture
{daniel.bedoya,lawrence.fyfe,elaine.chew}@ircam.fr

ABSTRACT

CosmoNote is a web-based application for visualizing and annotating expressive structures in performed music. This demo highlights the flexibility of CosmoNote's reconfigurable interface, features, and toolsets to design experiments to advance knowledge in music science, music perception, and music expressivity. CosmoNote's music pieces are organized into collections accessible by user accounts with defined roles. The music representations integrate synchronized visual layers representing recorded piano performance data and features to facilitate and inform annotations. The web app provides four annotation types: boundaries, regions, note-groups and comments. Through the selection of pieces, collections and user roles, CosmoNote's features can be tailored to investigate diverse research questions about experimental design for music audio-based annotations.

1. INTRODUCTION

Expressive structures in performed music exist beyond the score, being communicated by the performer through the addition of acoustic variations like intensity, timing, pauses, and articulation. Automatic recognition and extraction of expressive music structures poses major computing challenges. To tackle this problem, CosmoNote [1] recruits the volunteer thinking of human participants, called citizen scientists, to annotate these structures in performed music. While it is possible to annotate some expressive music structures using sound alone, in order to help annotators discern these expressive structures, CosmoNote provides layered visual representations of the music as cognitive scaffolding [2]. Experiments designed using CosmoNote aim to develop better strategies to increase annotators' awareness and appreciation of expressive nuances in recorded music performances. Here, we show how CosmoNote enables a variety of efficient study designs.

2. CONFIGURING COSMONOTE FOR A STUDY

CosmoNote's tools and visuals are designed to be highly re-configurable for performed music studies. CosmoNote's input data organized into three main types of documents: pieces, collections, and user roles.

Pieces: Music data, called pieces, are the core researcher input of CosmoNote. Each piece incorporates visual layers that can be shown or hidden by the annotators or researchers, depending on the experiment. The recorded performances are encoded as MP3 audio (plotted as a waveform) and/or MIDI data (displayed visually in a piano-roll style with notes and pedal information). Musical descriptors are computed from the audio and MIDI recordings as well as the scores for the performances. These data curves include loudness (derived from the audio recording), tempo (derived from the score and the MIDI recordings), and harmonic tension data (derived from the score). CosmoNote also allows the visualization of optional supplementary data (e.g., ECG signals) synchronized with the music. CosmoNote allows experimenters to present pieces in one of three modes: visuals with audio (default), audio only and visuals only. These modes extend the scope of studies that are possible with the app.

Collections: Pieces are shown to annotators in a set, called a collection. Pieces in a collection are chosen because they share individual properties (e.g., performer, composer) or an overall theme (e.g., music from the romantic era). Participants navigate through collections and complete annotation tasks that are specific to that collection. Each collection can include customizable options for: presentation (show/hide piece information), navigation (change piece order and piece selection), controls (show/hide buttons), visualizations (show/hide visual representations), annotation types (allow one or more types), and writing to the database (grant/revoke saving privileges). Pieces in a collection can be displayed in a fixed order, or can be shuffled for each participant, storing the presentation order for later analysis.

User roles: To access CosmoNote [1], annotators create a user account or are provided with one. User accounts allow annotators to have a personalized experience in which they can, for example, save their annotations between listening sessions. Each user account encompasses many associated properties (identification, account setup, among others), and is linked to data files containing all annotations marked by respective users in CosmoNote. For experimental design purposes, a user role property is associated with each account. User roles give experimenters the possibility to control precisely what type of access users have. There

[1] https://cosmonote.ircam.fr

Copyright: © 2022 Daniel Bedoya, Lawrence Fyfe, Elaine Chew et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

are three main types of roles: 1) super users, who can see all the content inside the CosmoNote server, which is hidden for everybody else; 2) public users, who can only see active public campaigns (this is the default role assigned when creating an account); and 3) custom users, who will see only a subset of the data that is different from the public campaigns (e.g., a set of pieces for a custom experiment).

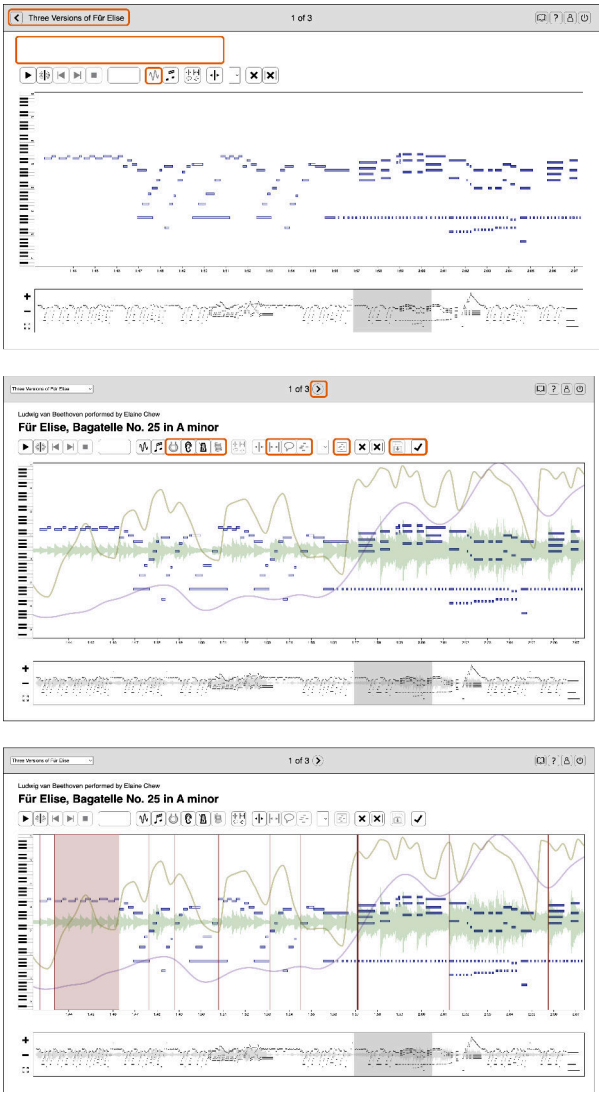


Figure 1. Variations of CosmoNote's interface in two experimental cases: Few options enabled (top) vs All options enabled (middle). Orange boxes highlight the differences. Boundary annotations of different strengths and regions are displayed in shades of red in the last panel (bottom)

3. ANNOTATING IN COSMONOTE

For the demonstration, attendees will be invited to participate in a CosmoNote annotation experiment. Through selected example cases, participants will learn how to actively listen for the music structures that arise from the decisions and actions of a performer, and see how these structures correlate with the music and audio features displayed visually on the CosmoNote interface.

Annotators will obtain/create a user account integrating a common user role and will be shown a sample of CosmoNote's collections. Pieces in each collection will contain different visuals to illustrate the information layer's use in specific contexts. To show CosmoNote's annotation capabilities, one visual layer has been created to show all four interactive annotation types: boundaries (vertical lines in time, with four levels), regions (time selections with start and end times), note-groups (selections of notes), and comments (miscellaneous indications).

The following examples are just a small selection of the configurations available in the CosmoNote interface. The example on Fig 1 shows two different configurations of CosmoNote's interface. Fig 1 (top) is a restricted version of the app: Only two visualizations are possible (waveform –toggled off– and piano-roll), only boundary annotations can be placed, and navigation is forced (users cannot advance to the next piece until they finish annotating the current piece). In contrast, Fig 1 (middle) shows a complete version of the app: All visualizations, annotation types and navigation options are available to the user. Fig 1 (bottom) shows boundaries (red vertical lines) and regions (light-red rectangle) layered on top of the music data and feature visualizations (pedal data and harmonic tension are toggled off).

4. SUMMARY

CosmoNote was created for enabling people with diverse backgrounds to mark structures such as boundaries, accents, and salient melodies that they hear and experience while listening to recordings of performed music. Demo participants will learn about the conception and design of CosmoNote and its principal annotation methods. They will also learn about the possibilities for study design using this new tool. The demonstration is conceived to showcase the capabilities of CosmoNote and the study design options so as to equip annotators with the knowledge to make the most of the web app while creating an environment that fosters constructive feedback for the evolving design and use of CosmoNote.

Acknowledgments

This result is part of the project COSMOS that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement Nº 788960).

5. REFERENCES

[1] L. Fyfe, D. Bedoya, C. Guichaoua, and E. Chew, "Cosmonote: A web-based citizen science tool for annotating music performances," in *Proceedings of the International Web Audio Conference*, 2021.

[2] N. Yelland and J. Masters, "Rethinking scaffolding in the information age," *Computers & Education*, vol. 48, no. 3, pp. 362–382, 2007.

Calliope: Generating Symbolic Multi-Track Music on the Web

Renaud Bougueng
Simon Fraser University
rbouguen@sfu.ca

Jeff Ens
Simon Fraser University
jeff_ens@sfu.ca

Philippe Pasquier
Simon Fraser University
pasquier@sfu.ca

1. EXTENDED ABSTRACT

The development of computer-assisted composition (CAC) systems is a research activity that dates back to at least the works by IRCAM on OpenMusic [1]. CAC is a field that is concerned with developing systems that are capable of automating partially or completely the process of music composition. There exists several compositional tasks a system can address (e.g. rhythm generation, harmonization, melody generation, etc). These tasks can be realized with machine learning (ML) algorithms given a conditioning or not on prior musical sequences. Many ML-based CAC systems have emerged from both academia and industry over the years [2] [3]. For the majority of them, the user continuously generate music by tweaking a set of parameters that influences the model's generation.

Building on top of Apollo, an interactive web environment that makes corpus-based music algorithms available for training and generation via a convenient graphical interface [4], Calliope is specialized for advanced MIDI manipulation in the browser and generative controllability of the Multi-Track Music Machine (MMM) model [5] for batch generation of partial or complete multi-track compositions. The aim is to enable the ability for composers to effectively co-create with a generative system. Calliope is built in Node.js, the Web stack (HTML, CSS, Javascript) and MongoDB. It is made interoperable with the MMM pre-trained model via the Python runtime.

MMM offers both global-level deep learning parameters (e.g. temperature) and track-level music-based constraint parameters: note density, polyphony range and note duration range. Bar selection can be used to refine the request for generation. It is also possible to delete or add MIDI tracks to an existing MIDI file in order to generate on a subset of the tracks or to generate a new track for a given composition. The composer makes use of all these varied controls to steer the generative behavior of the model and guide the composition process. Batch generation of musical outputs is implemented via the MMM's Python interface which offers batch support natively. This ability means that the composer can rapidly explore alternatives, including generating from a previously generated output, for a given set of control parameters. We have tested batch requests of 5, up to 1000 generated music excerpts at a time, which take from 3 seconds to 10 minutes on an average computer depending on the note density of the music input. This process drives an interactive loop of continuous music generation and playback listening for the composer to navigate the creative process.

Copyright: © 2022 Renaud Bougueng et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

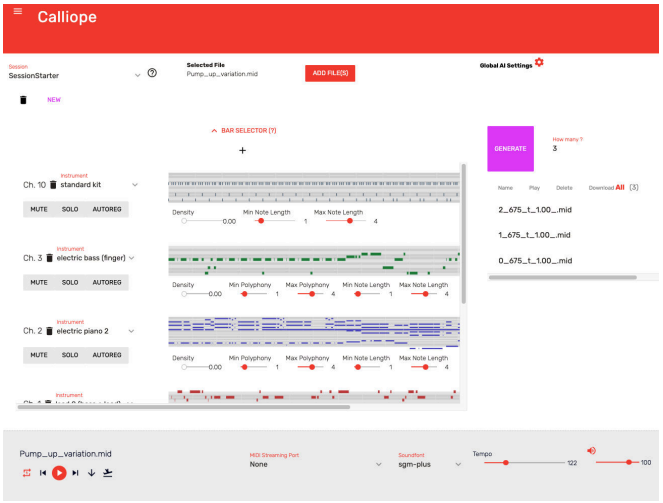


Figure 1. Calliope's Interface

2. REFERENCES

[1] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue, "Computer-assisted composition at ircam: From patchwork to openmusic," *Computer Music Journal*, vol. 23, no. 3, pp. 59–72, 1999.

[2] C. Anderson, A. Eigenfeldt, and P. Pasquier, "The generative electronic dance music algorithmic system (gedmas)," in *In Proceedings of the Second International Workshop on Musical Metacreation (MUME 2013) 2013.*, 2013.

[3] A. Roberts, J. Engel, Y. Mann, J. Gillick, C. Kayacik, S. Nørly, M. Dinculescu, C. Radebaugh, C. Hawthorne, and D. Eck, "Magenta studio: Augmenting creativity with deep learning in ableton live," 2019.

[4] R. B. Tchemeube, J. Ens, and P. Pasquier, "Apollo: An interactive environment for generating symbolic musical phrases using corpus-based style imitation," 2019.

[5] J. Ens and P. Pasquier, "Mmm: Exploring conditional multi-track music generation with the transformer," *arXiv preprint arXiv:2008.06048*, 2020.

AN INTERACTIVE MUSIC INFILLING INTERFACE FOR POP MUSIC COMPOSITION

Rui Guo
University of Sussex
R.Guo@sussex.ac.uk

ABSTRACT

Artificial intelligence (AI) has been widely applied to music generation topics such as continuation, melody/harmony generation, genre transfer and music infilling application. Although with the burst interest to apply AI to music, there are still few interfaces for the musicians to take advantage of the latest progress of the AI technology. This makes those tools less valuable in practice and harder to find its advantage/drawbacks without utilizing them in the real scenario. This work builds a max patch for interactive music infilling application with different levels of control, including track density/polyphony/occupation rate and bar tonal tension control. The user can select the melody/bass/harmony track as the infilling content up to 16 bars. The infilling algorithm is based on the author's previous work, and the interface sends/receives messages to the AI system hosted in the cloud. This interface lowers the barrier of AI technology and can generate different variations of the selected content. Those results can give several alternatives to the musicians' composition, and the interactive process realizes the value of the AI infilling system.

1. INTRODUCTION

Most of the work on AI music generation provides a google colab notebook for exploration [1]. That notebook hides the technical details and makes it easier to control the system, however, it is still far from an intuitive user experience. The survey [2] shows the musicians prefer more control rather than generating end-end music. The survey suggests there are several gaps between musicians and those algorithms, in the concept of "disposable", "context-aware" and "steerable".

To fill in that gap, this work builds a Max patch integrated with an AI pop music infilling model with multiple level control parameters. Given a MIDI file with one to three tracks in the types of melody, bass and harmony, the user can select a whole track/several bars and have new variations given the surrounding music, which is to have an "infilling" of the selected music region. The track/bar control parameters can help the user to steer the music to explore different textures, and tonal tension control can change the total tension curve by drawing a line. The user can compose a pop song by interactively repeating the generation process in this interface with ease.

Copyright: © 2022 Rui Guo et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. MOTIVATION

This work is based on a pop music infilling system with multiple levels of control [3]. That system can steer each track's note density/polyphony/occupation level and bar tonal tension [4]. The original interface in [3] is a colab notebook, and to make it more accessible, a Max patch is prepared to communicate with the deep learning infilling system and better fit the composer's writing process. Compared with other interfaces [5], this one provides both track and bar control. It is also disposable and context-aware by the nature of music infilling. The aim of this interface is to be compatible with any music DAW that can receive MIDI messages, and easy to set the infilling region and control parameters. It should also play/display the music notation in the interface. The target user of this interface is the music composers of all levels.

3. INTERFACE

The input of this interface is MIDI files and it will use the first three tracks as input to the system. The tracks should be in the types of melody, bass and harmony with arbitrary order, but each type can only be in one track. After clicking the "Load a file" button in Figure 1, the MIDI information including track number, bar number, metre, tempo and key are displayed. The key calculation is based on the tonal tension theory and also the music21 [6] key estimation.

Before the infilling process, each track's type should be set to one of the "melody", "bass", "harmony" or "empty". Because the input music length can be arbitrary, the user needs to set a start bar of music if the total bar length is more than 16, which is the bar limit of the proposed infilling system. To process the whole song, the user can apply this tool to any 16 bar sections of the whole song. After clicking the "Calculating control" button, the track/bar control parameters of the region from the selected start bar are calculated and displayed in the Figure 2.

The infilling process will assume the music has three tracks. If the input music has less than three tracks, the empty track can also be infilled in the infilling stage. The minimum infilling region of this application is one track in a bar, e.g. melody track in bar 5. A whole bar can be infilled by setting "infilling tracks" to "all". Each track has its own control, and each bar tonal tension control can also

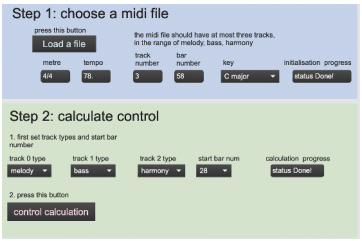


Figure 1. Input MIDI selection and track type settings

be altered. The track parameters are in the range of level 0-9, and the bar tonal tension is set by dragging the slider according to that bar. The user can also drag a tension curve and infill those selected bars.

After clicking the "start infilling" button, the return result can be played/saved by the play control in Figure 3. The bach [7] library is employed for music notation display and each track is in a separate music sheet. The user can send the tracks to different DAWs by double clicking the "midiout" button.

The node.script module is utilised in Max to communicate with a PyTorch model served in the Flask framework. Only the MIDI file and the control information is sent/received with little bandwidth requirement. The model is served in cloud with GPU and can be also installed on private servers if needed. The Max patch is shared for free exploration [1]

4. INFILLING EXAMPLES

A starting point can be a complete MIDI file with three tracks and at least 16 bars. The infilling function is applied iteratively to change part of the music. From the author's experience, the infilling result is better if working on a smaller scale with more surrounding context, e.g. to infill 8 bars of melody or 4 whole bars at the beginning of music. If set the "go back to last result" to "No", the next infilling operation will be based on this result.

A MIDI with less than three tracks and only a few bars can also be used to build an entire song. If the input music has only a melody track, by selecting the control parameters of the bass/harmony tracks, those tracks can also be infilled according to the control. Due to the sampling operation in the generation, the generated result generally follows the control, but it is not ensured the result control level is always the same as the settings.

5. CONCLUSIONS

This work builds a Max patch to integrate an AI music infilling model with an intuitive interface. The infilling region can range from a track in a bar to the whole song. The track/bar level control helps to steer the generation process, and the generated MIDI can be sent to different DAWs for further edit. This tool can give numerous variations of the input and inspire the composer in the music writing.

[1] https://github.com/ruiguo-bio/max_infilling

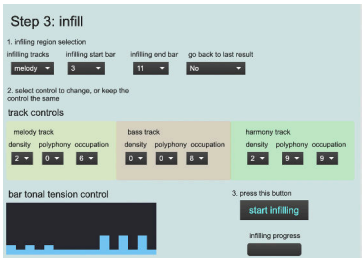


Figure 2. Music infilling settings. Arbitrary sections in the music can be selected for infill, and the track/bar control parameters can be adjusted.

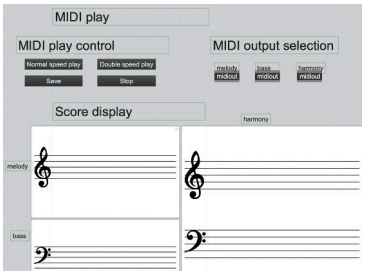


Figure 3. The play control and music notation display. The result MIDI can be sent to different DAWs for further edit or saved. Each track is in a separate music sheet.

6. REFERENCES

[1] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[2] C.-Z. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. J. Cai, "Ai song contest: Human-ai co-creation in songwriting," *ArXiv*, vol. abs/2010.05388, 2020.

[3] R. Guo, I. Simpson, C. Kiefer, T. Magnusson, and D. Herremans, "Musiac: An extensible generative framework for music infilling applications with multi-level control," *arXiv preprint arXiv:2202.05528*, 2022.

[4] D. Herremans and E. Chew, "Tension ribbons: Quantifying and visualising tonal tension," in *2nd Int. Conf. on Technologies for Music Notation and Representation*, Cambridge, UK, 2016, pp. 8–18.

[5] G. Hadjeres and L. Crestel, "The piano inpainting application," *arXiv preprint arXiv:2107.05944*, 2021.

[6] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data," in *ISMIR*. International Society for Music Information Retrieval, 2010, pp. 637–642.

[7] A. Agostini and D. Ghisi, "A max library for musical notation and computer-aided composition," *Computer Music Journal*, vol. 39, no. 2, pp. 11–27, 2015.

Reconstructing Prehistoric Music Technologies: Archaeological Explorations of Humans as Designers

Miriam A. Kolar
Center for Computer Research in Music and Acoustics (CCRMA), Stanford University
kolar@ccrma.stanford.edu

DEMO ABSTRACT

Despite ubiquitous references to human curiosity and experimentalism across cultures, narratives about life in prehistory (before writing) tend to oversimplify our physiologically equal ancestors. Compounding the tendency to primitivize ancient humans and discount their experimental and expressive capacities is the archaeological practice of material categorization, which has privileged singular definitions of artifact objects. Far more likely than such neat explanations is the proposition that humans encountered sonic properties of materials, objects, and places while in the midst of non-musical activities — and noticing sounds, found them to be compelling and useful in particular ways. As in the present, prehistorical experimentation with how things work to make and shape sound — engineering design — would have led to the creation and refinement of sound-making tools (mechanical music technologies), and also to the use and modification of environmental settings with notable acoustical reinforcement and thus generative of sonic sensory feedback.

Sound and music computing offers an expanding range of tools for studying and demonstrating musical techne in archaeological research. This demo draws on case-study research in the Peruvian Andes and southern France to show how performance studies, acoustics, and auditory science methodologies bring present-day music technologies into sensory conversation with the traces of musical experimentation millennia past. Beyond the scientific documentation of artifact instrument performance, digitized replications, and virtual reconstructions of sounds and sounding contexts, computational methods are enabling new forms of research to parse and represent the archaeological possibility space of artifact materials of ancient musical technologies, revealing the ways that ancient humans were as much designers as we imagine ourselves today.

Acknowledgments

Thanks to my many collaborators in research about the 3,000-year-old conch-shell horns of Chavín de Huántar, Perú, a UNESCO World Heritage Centre archaeological site where seashells sounded across the Andes to connect the coast with the rainforest in cosmological terms (https://ccrma.stanford.edu/groups/chavin/team.html). I would not be working in archaeoacoustics nor on research in Paleolithic decorated caves were it not for vision and organizational efforts of John Chowning, who has assembled a group of extraordinary colleagues for the Paleo-Acoustics project with archaeologist Carole Fritz, prehistorian Gilles Tosello, and an inspiring pluridisciplinary team. Special thanks to Luna Valentin and Romain Michon for getting us into speleoacoustics in Ardèche.

REFERENCES

[1] M. A. Kolar, R. Ramírez Rodríguez, R. Guerrero de Luna Rueda, O. Silva Espinoza, R. San Miguel Fernández, "Experimentos acústicos y estudios de actuación con los pututus de Chavín de Huántar: Evidencias de su expresividad en relaciones entre seres diversos en un centro ceremonial del periodo Formativo," presented at the Primer Congreso Internacional de Etno y Arqueomusicología, July 2021.

[2] M. A. Kolar, "Conch Calls into the Anthropocene: Pututus as Instruments of Human-Environmental Relations at Monumental Chavín," *Yale Journal of Music & Religion*, 2019, 5(2), pp.22-63, https://doi.org/10.17132/2377-231X.1151.

[3] M. A. Kolar, R. A. Covey, and J. L. Cruzado Coronel, "The Huánuco Pampa acoustical field survey: an efficient, comparative archaeoacoustical method for studying sonic communication dynamics," *Heritage Science* 6(39), https://doi.org/10.1186/s40494-018-0203-4.

[4] M. A. Kolar, "Pututus, Resonance and Beats: Acoustic Wave Interference Effects at Ancient Chavín de Huántar, Perú," presented at the Acoustical Soc. of America 168th Mtng., Indianapolis, Oct. 2014: https://ccrma.stanford.edu/groups/chavin/ASA2014.html.

[5] L. Valentin, M. Kolar, and P. Monteil, "Speleoacoustics in Southern Ardèche for Auralizations and Music Experimentation," in Proc. Int. Conf. Sound and Music Computing (SMC2022), Saint-Étienne, 2022, pp.TBD.

Networked Performances with Ossia Score

Jean-Michaël Celerier
ossia.io
jcelerier@ossia.io

Pia Baltazar
ossia.io
pia@ossia.io

This demo will present the current work-in-progress in the ossia community towards authoring and performing networked artworks. The demo is based on the free and open-source software ossia score which recently reached version 3. Dubbed "interactive sequencer for the intermedia arts", it combines both the non-linear time-lines and the data-flow paradigms to allow artists to create rich evolving multimedia artworks, musical pieces, museum installations, etc.

1. INTRODUCTION

ossia score is a free and open-source sequencer available for macOS, Windows, desktop and embedded Linux, with an in-development web-based version. The software can be downloaded at `ossia.io`.

Version 3 introduces support for a real-time GPU-enabled video pipeline seamlessly integrated with the rest of the system, live-coding for C++ (and many other languages), support for tempo, musical metrics, hierarchical polyrythms, and a generalized looping primitive, among other features.

An experimental branch of the software had been started in 2016 to explore the domain space of networked and distributed artworks, at both the authoring and performance stages: this work had been presented at the JIM [1].

This demo is an update of this work, which tries to explore the capabilities of the networked authoring and execution features in the context of a small experimental artwork showcasing how the feature can be used to share behaviours across multiple computers.

2. DEMO EXPLANATION

The demo implements a scenario which plays distributed sound and video content in a synchronised way: the score defines on which computer which part of the score runs. Multiple computers can join the session and do their part of the performance in sync.

This is done through specific annotations that were implemented in ossia score: the composer can tag a specific part of the score as pertaining to a group and another part to another group. Computers can then connect to the network session and assign themselves to one or multiple groups.

The demo will also showcase how distributed edition works. The source code is available at `https://github.com/ossia/iscore-addon-network`.

Copyright: © 2022 Jean-Michaël Celerier et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Figure 1: An example score in *ossia score*

3. FURTHER GOALS AND RESEARCH IDEAS

3.1 Exchanging Data

The original distribution work only covered the introduction of distributed semantics for the temporal execution of the system. Current work is ongoing to ensure that all the data types supported by ossia score—basic messages, but also audio and video streams—can be seamlessly exchanged across applications and systems to ease the production of large-scale distributed artworks.

3.2 Towards Distributed Live Performances?

The ossia project is interested in studying not only the technical and semantical features of distributed artwork authoring and performance, but also the artistic and social implications of those artworks: how can such systems allow more people to experience the act of going to a concert or live performance in a social context impacted or disrupted by various factors, such as climate change, pandemies, etc.? Can we qualify and quantify what going to a live performance involving new media entails, and how closer we can get to the original experience in the context of a world which has never come closer to Asimov's isolated world depicted in *The Naked Sun* than in the last few years, and where arising ecological constraints will keep the pressure high on reducing physical transportation for a long time? How can we create distributed performances happening simultaneously and connectedly in various physical places, and how can the performance's sense of spontaneous community be transposed in this context?

THE ACCEPTABLE RANGE OF PITCH FREQUENCIES AND DURATION OF NOTES IN SINGING

Behnam Faghih
Dept of Computer Science,
Maynooth University,
Maynooth,
Co. Kildare,
Ireland
Behnam.Faghih@mu.ie

Joseph Timoney
Dept of Computer Science,
Maynooth University,
Maynooth,
Co. Kildare,
Ireland
Joseph.Timoney@mu.ie

ABSTRACT

Performing musical notes correctly does not mean that all the performers play the notes in the exact same pitch, onset, offset, duration, and loudness, but they normally perform the notes within acceptable psychoacoustic ranges.

Sundberg et al. [1] studied what mean F0s were accepted as being "in tune" and "out of tune". The results showed that for most of the tones deemed to be in tune they had an average F0 that varied within a narrow band of about ±7 cents, whereas most tones judged as being out of tune were outside this frequency band. Moreover, Sundberg et al. [1] found that singers exhibited the same patterns of changing intonation when performing the same notes but using slightly different frequencies when they repeated these notes in other bars.

According to the Seashore [2] study, long notes were sung with an average F0 that coincides with the theoretically correct value. Moreover, many of the long tones were observed to change their average frequency in various ways during the tone. Bjørklund [3] found that such deviations were typical for professional singers as opposed to nonprofessional singers. With regard to short tones, the relationship between F0 and the theoretical pitch seems to be considerably more complicated [4].

Sundberg & La [5] analysed the tuning of premier baritone singers and found examples of quite large deviations from equal-tempered tuning, sometimes exceeding 50 cents. In particular, the highest note in phrases with an agitated emotional character was often sharpened. The intonation of such tones was flattened to equal-tempered tuning, and a listening test was run in which musician listeners were asked to rate the expressiveness in a pair-wise comparison between the original version and the version with manipulated tuning. There was a significant preference for the original versions. This result indicates that intonation can be used as an expressive device in singing.

Based on the above explanation, researchers have found that the amount of allowable/imperceptible frequency deviation in each sung note depends on its position in a piece of music. However, since there are some limitations in the previous studies, the exactly acceptable ranges of pitch, onset, offset, duration, and loudness of a note in a piece of music is not fully understood yet. Therefore, this study examines a dataset of recorded vocals to discover the true relationship regarding the acceptability between the perceived F0 and duration of a note against its theoretical frequency and duration. As the result, this study provides an algorithm to calculate the acceptable range of frequencies and duration of each note based on its position in a piece of music. Therefore, the algorithm uses a data driven approach to derive the rules that are implemented in the new software algorithm.

Keywords: singing, tune, pitch frequency, duration, acceptable range.

REFERENCES

[1] J. Sundberg, E. Prame, and J. Iwarsson, "Replicability and accuracy of pitch patterns in professional singers," *STL-QPSR*, vol. 2–3, pp. 51–62, 1995.

[2] C. E. Seashore, *Psychology of Music*. New York: Dover, 1967.

[3] A. Bjørklund, "Analyses of Soprano Voices," *J. Acoust. Soc. Am.*, vol. 33, no. 5, pp. 575–582, May 1961.

[4] J. Sundberg, "Perception of Singing," in *The Psychology of Music*, Third Edit., no. 1960, Elsevier, 2013, pp. 69–105.

[5] J. Sundberg and F. La, "Is Intonation Expressive?," in *40th Annual Symposium on Care of the Professional Voice*, 2011.

THE SPRINGAMAJIG: A FLEXIBLE DIGITAL MUSICAL INSTRUMENT

Thomas Rushton
Aalborg Universitet, København
A. C. Meyers Vænge 15
2450 København SV, Denmark
trusht21@student.aau.dk

Dan Overholt
Aalborg Universitet, København
A. C. Meyers Vænge 15
2450 København SV, Denmark
dano@create.aau.dk

ABSTRACT

The Springamajig is a prototype gestural digital musical instrument based on the physical properties of a large helical extension spring suspended between two long handles. Equipped with a small collection of sensors which measure its physical orientation and manipulation, plus a microphone and speaker, the instrument is self-contained, and sonifies gestures by way of a bespoke, granular-type table-lookup synthesis engine running on an internal microcontroller.

The instrument has the potential to present an engaging and compelling experience for performer and audience, respectively; this is due in part to the resistance provided by the spring under deformation, and the physical effort required in order to perform with it. Its physical properties lend it to a variety of performance approaches, taking advantage of its weight and momentum.

1. INTRODUCTION

The physicality of digital musical instruments (DMI) is a topic that has been given serious consideration within the field of New Interfaces for Music Expression [1]. Arguably the physical *dialogue* that one enters into with a traditional musical instrument is a key aspect in making playing that instrument compelling.

The Springamajig seeks to incorporate the important sense of effortfulness in performance (and in observing that performance), while also emulating the *standalone* nature of traditional music instruments. In terms of DMI taxonomy it represents an *alternate controller*, but, given the inspiration drawn from embodied and gestural controllers such as the *T-Stick* [2] and *Accordiatron* [3], it is fair to ascribe a degree of instrument-like control [4].

2. DESIGN AND IMPLEMENTATION

2.1 Construction

The instrument is constructed around its central spring. The metal inner handle at each end of the spring is surrounded by a $40\,\mathrm{mm}$ diameter polypropylene tube, $1\,\mathrm{m}$ in length. One outer handle, designated *active*, terminates in a box containing a Teensy 4.0 microcontroller [1] to which sensor data is relayed, a small amplifier circuit, a 3" loudspeaker, and separate batteries to power the amplifier and microcontroller. A single power switch is installed on the outside of the box.

2.1.1 Sensors

A 4.5" flex sensor is installed within the hollow part of the spring, and effectively registers the amount of curvature in the spring. A force sensitive resistor (FSR) is situated between the inside of the active handle and the outside of the corresponding inner handle and detects changes in the squeezing force between the two. A 3-axis digital accelerometer (LIS3DHTR) is mounted to a foam plug inserted into the active handle and reports its spatial orientation along x, y and z axes.

An I2S MEMS microphone (SPH0645LM4H) on a breakout board is attached to the inner handle at the point at which the active handle terminates near the spring.The microphone picks up the sound of the spring creaking under manipulation where it joins the inner handles. It is also sensitive to ambient sound, and the performer can achieve interesting effects by singing or whistling into the microphone. A diagram of the construction of the instrument can be seen in Figure 1.

2.2 Sonification

Observing the *granular* nature of the creaking sounds produced by the spring at its mounting points, the development of a granular synthesis engine in the Faust [2] audio programming language was conducted. Inspiration was found in Mykle Hansen's *Weather Organ* [5], a system for generating soundscapes based on stochastically distributed "sparse noise". Weather Organ's `sparse_periodic_trigger` function was adapted to trigger the generation of a windowed grain from a sample lookup table. Faust code was compiled for the Teensy microcontroller and combined with a simple routing algorithm that combined a small proportion of the unprocessed microphone input with the output of the granular synthesis implementation [3].

Copyright: © 2022 Thomas Rushton et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[1] https://www.pjrc.com/store/teensy40.html
[2] https://faust.grame.fr/
[3] Code can be found at https://github.com/hatchjaw/springamajig

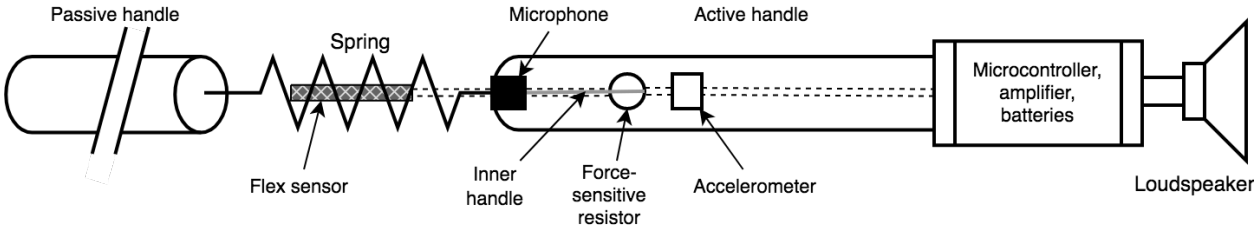
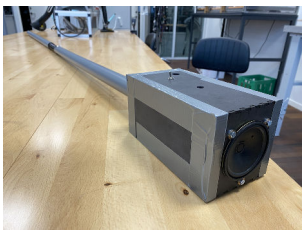


Figure 1: Diagram of the instrument: Dashed lines represent internal wiring, connecting sensors to the microcontroller. The force-sensitive resistor lies between the outer face of the right-hand inner handle and the inner face of the active handle. The microphone is situated atop the inner handle, accommodated by a notch cut into the end of the left extremity of the active handle; the microphone is oriented with its port facing the inner handle.

(a) Detail of spring and embedded microphone



(b) Detail of box enclosure and speaker

Figure 2: Images of completed prototype of *The Springamajig*.

2.3 Mapping

The primary mapping relates to the flex sensor embedded within the spring. The value of this sensor is mapped to three distinct aspects of the granular synthesis engine: 1) Grain density is proportional to the amount of flex. 2) Flex is also mapped proportionally to grain duration, within the range $0.005\,\mathrm{s} < t_{\mathrm{grain}} < 0.5\,\mathrm{s}$. 3) Above an arbitrary threshold (62.5% of the maximum), further samples are no longer written to the grain lookup table. Performers can thereby flex and *hold*, then change the orientation of the instrument to alter, for example, the grain playback rate.

The gesture required to change the amount of pressure on the FSR is quite subtle, therefore it is mapped to a relatively subtle parameter, that of *grain regularity*. Low FSR values result in irregular grain distribution; high values result in regular distributions that can give rise to periodic, rhythmic effects.

The z-axis of the accelerometer is mapped to the grain playback rate. Inclining the active handle will cause the grain playback rate to increase, gradually, to maximum of 200%; declining it will result in the rate slowing and eventually reversing, to a 'minimum' of -200%.

The x- and y-axes of the accelerometer correspond to rotations along the roll axis of the instrument. For y, the absolute sensor value is taken, and used to scale the 'room size' and gain of a reverberation algorithm applied to the output of the granular synthesis engine. If the absolute value of x undercuts the normalised flex sensor value, x will override the grain duration. x can be used to affect output texture by modulating the grain duration during performed gestures incorporating high levels of flex.

3. CONCLUSION & PERFORMANCE METHODS

The Springamajig represents a novel DMI exhibiting a potentially compelling experience for both performer and audience. The 'standard' way to wield *The Springamajig* is by its handles, somewhat like grasping a pair of garden shears. As the performer brings their hands together, the spring bends and the microphone detects its creaking (in addition to sampling any other sounds the performer makes), which are fed into the granular synthesis engine. The performer may now raise or lower the spring to change the grain density and playback rate, perhaps hooking the handles under their arms for greater control. A particularly enjoyable technique involves holding the instrument vertically by the passive end, and allowing the active end to drop under its own weight, then guiding it around as it bounces and spins overhead. The performer has less direct control in this scenario, but can *encourage* the instrument into variations in grain texture. Initial feedback from test performers is promising, but it should be subjected to evaluation by a variety of performers in differing conditions. Finally, the instrument's sonic fidelity could be improved with better speakers (and/or sending the audio wirelessly).

4. REFERENCES

[1] P. Bennett, N. Ward, S. O'Modhrain, and P. Rebelo, "Damper: a platform for effortful interface development," in *Proceedings of the 7th international conference on New interfaces for musical expression*, 2007, pp. 273–276.

[2] J. Malloch and M. M. Wanderley, "The t-stick: From musical interface to musical instrument," in *Proceedings of the 7th international conference on New interfaces for musical expression*, 2007, pp. 66–70.

[3] M. Gurevich and S. von Muehlen, "The accordiatron: A midi controller for interactive music," *arXiv preprint arXiv:2010.01574*, 2020.

[4] D. Overholt, "Musical interface technology: Multimodal control of multidimensional parameter spaces for electroacoustic music performance," 2007.

[5] M. Hansen, "Weather organ: A sparse stochastic synthesizer in faust," in *Proceedings of the 1st International Faust Conference (IFC-18)*, 2018.

Participation in the SMC-2022 conference was supported by the European Art Science and Technology Network (EASTN-DC)

Demos: Støjbox XYZ and Støjbox Spring

Benjamin Fullerton Støier
Aalborg University, Copenhagen
bstoie21@student.aau.dk

Dan Overholt
Aalborg University, Copenhagen
dano@create.aau.dk

ABSTRACT

This paper describes the features, development process, and demonstration of two prototypes of new desktop instruments: Støjbox XYZ, and Støjbox Spring. The highlight of the Støjbox XYZ is a 3-dimensional controller designed to augment existing sensor technologies, thereby increasing the expressiveness of the synthesizer interface by letting musicians simultaneously access three different sound parameters with a single finger. The main element of the Støjbox Spring is a spring reverb tank that can be played directly (with the addition of an internal feedback path incorporating various effects), or used as an effect unit for other instruments. The method of iterative prototyping was used in the development of these instruments, and the work contributes with the bespoke designs and demonstrations of the prototypes.

1. INTRODUCTION

Figure 1. Støjbox XYZ

Støjbox XYZ is a desktop synthesizer designed with the goal of allowing for more expressive control over the sound parameters. It has an interface that features an XYZ controller, allowing users to simultaneously control multiple sound parameters by moving fingers in the X-, Y- and Z-directions. It also includes a menu system that can be navigated using an encoder and push buttons, allowing the aforementioned XYZ controller to be dynamically mapped to the sound parameters. The menu system also features an LFO section that allows the user to select the LFO shape and the LFO mapping. Finally the menu allows for control over effects and provides access to a pitch envelope. Other features include 16 knobs allowing for more traditional synthesizer interactions. Støjbox XYZ uses these 16 knobs to alter the following fixed parameters: oscillator, filter, LFO, and the ADSR envelopes of amplitude and filter cutoff.

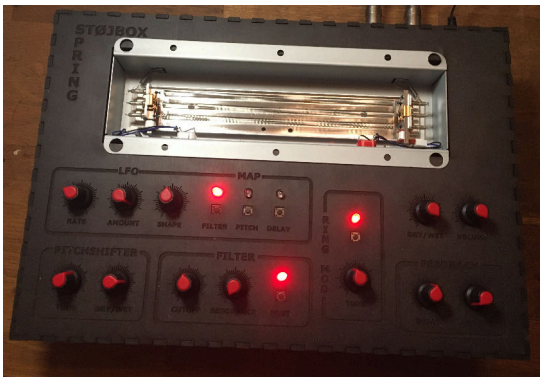


Figure 2. Støjbox Spring

Støjbox Spring extends upon the standard spring reverb effect, by allowing the output of the spring to be fed back in to the input of the spring reverb, though additional digital effects. Spring reverbs on their own have a fixed decay time that depends on the size of the spring reverb tank used. However, by applying feedback the decay time can be increased. When increasing the amount of feedback, the spring reverb will eventually start to self oscillate, and thus create a reverb with an infinite decay time. The Støjbox Spring allows for the possibility to animate the reverb, augmenting it by applying various effects to the feedback path. With these features and the direct spring manipulation in mind, the Støjbox Spring can't be regarded as just an audio effect, but is instead an expressive feedback instrument in its own right.

2. SOUND PROCESSING

The processor used in the Støjbox XYZ and the Støjbox Spring is the Daisy Seed, an embedded platform for music creation, created by Electro-Smith. The Daisy Seed is a small board with an ARM Cortex-M7 MCU, running at 480MHz with an external 64MB of SDRAM added for audio memory [1].

The Støjbox XYZ supports 16 voices of polyphony and uses a variable waveshape oscillator for sound generation. The oscillator has three parameters, besides the pitch, that

Copyright: © 2022 Benjamin Fullerton Støier et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

can be controlled. The first is waveshape, which sets the waveform to either a square, or a "saw/tri/ramp" waveform (which can be interpolated between these). The second oscillator parameter is the pulse width, which sets the pulse width of the square waveform, or determines whether the "saw/tri/ramp" waveform takes shape as a sawtooth, triangular or ramp waveform. The final parameter sets the frequency of the sync oscillator. Støjbox XYZ incorporates a lowpass filter, allowing the user to have control over the cutoff frequency and resonance of the filter. The Støjbox has three ADSR envelopes dedicated to the amplitude, filter and pitch. The Støjbox XYZ also features an LFO, with the user having control over the rate, amount, mapping and shape of the LFO. Finally, the Støjbox XYZ also includes three effects that can be applied to the sound. These effects include reverb, chorus and overdrive. The order of the effect chain is overdrive first, then chorus, and finally reverb.

The Støjbox Spring features a spring reverb tank that can be fed back in to itself. The level of feedback can be adjusted, as well as the delay length (in number of samples). Additionally, the feedback signal can be manipulated by several effects, including a pitch shifter, a ring modulator, and a low pass filter. The pitch shift effect allows users to transpose the pitch of the feedback signal in the range of two octaves (one up, one down). It also features dry/wet control, allowing users to mix the pitch shifted signal with the original signal. One of the pushbuttons allows a ring modulation effect to be applied to the feedback signal, by multiplying the feedback signal with a sine wave oscillator (tuned in the range from 50 Hz to 5 kHz). The filter used in the Støjbox Spring is a Moog ladder style lowpass filter, and provides fixed controls for both cutoff and resonance. The filter can be applied to the output of the spring reverb tank instead of the feedback signal by pressing another button. The Støjbox Spring also includes an LFO that can be mapped to one or multiple parameters including the filter cutoff, the pitch shifter and the feedback delay. Users can control the amount, rate and the shape (sine, square, sawtooth or ramp) of this LFO.

The Støjbox spring can be used as an insert effect, and has wet/dry control allowing the user to mix the input of the Støjbox Spring with the output of the spring reverb tank. In this way, it can be used on its own as a feedback instrument (without any input signal), by setting the wet/dry control to 100% wet. The springs in the reverb tank are exposed to the user, which allows for interacting and "playing" them by striking or grabbing. The Støjbox spring interface includes 12 knobs and five buttons. Each of the five buttons has a dedicated red LED indicating the current state of the buttons. Being a feedback instrument, the produced sound can quickly become chaotic and unpredictable, therefore simple one-to-one mappings were chosen for all the knobs, in order to allow for some predictability when controlling the sounds.

2.1 XYZ controller

The XYZ controller uses the Trill Square by Bela as an XY pad. The Trill Square is a square shaped capacitive touch sensor capable of sensing a two dimensional position from a single touch. The Trill Square was designed for the Bela platform, but it works with any platform that supports communication through I2C, such as the Daisy [2]. In order to get the Z-dimension of the XYZ controller, a ratiometric linear hall-effect sensor a magnet were utilized. Kitchen sponges were used for the spring mechanism, as they proved to provide the best tactile response of readily available materials.

3. CONCLUSION

Støjbox XYZ is a new desktop synthesizer with an interface designed for the purpose of allowing augmented expression. The additional expression was incorporated by designing an XYZ controller. Custom mapping of the XYZ controller was an important feature for the authors, since it allows for more freedom and offers users to express themselves through their mapping design choices.

In order to create the custom mappings, a menu system was implemented. This menu system also allows for more functionality, such as selecting LFO mapping and LFO shapes, limiting parameters, applying effects, and adding a pitch envelope. Støjbox XYZ has proven to fulfill its main goals, but there is still plenty of room for future improvements in the areas of synthesis, and design improvements to reduce friction.

The Støjbox Spring extends on the well known spring reverb effect by allowing the output of the spring reverb to be fed back into its input through various effects, and thus contributes to the ever-growing family of feedback instruments [3]. The Støjbox Spring can be used as a standalone feedback instrument, but also offers the possibility to be used as an insert effect.

We have outlined the features and development process of the Støjbox XYZ and the Støjbox Spring in this work, while also explaining the authors' motivation and concepts for these two prototype instruments. Initial testing points towards both being very engaging to play, and while somewhat similar in appearance, they have vastly different sonic outputs. Readers are encouraged to see video documentation of the instruments in use, at the following web link: `https://tinyurl.com/5ewncs52`

This work has been partially supported by the EASTN-DC project (European Art-Science- Technology Network for Digital Creativity).

4. REFERENCES

[1] Electro-Smith Daisy Seed. [Online]. Available: https://www.electro-smith.com/daisy/daisy

[2] Bela Trill Square. [Online]. Available: https://shop.bela.io/products/trill-square

[3] A. Eldridge, C. Kiefer, D. Overholt, and H. Ulfarsson, "Self-resonating vibrotactile feedback instruments ||: Making, playing, conceptualising :||," 6 2021, https://nime.pubpub.org/pub/6mhrjiqt. [Online]. Available: https://nime.pubpub.org/pub/6mhrjiqt

KUPLEN: A HANDS-ON PHYSICAL MODEL

Gabriel Vincent Karl Gustafsson
Aalborg Universitet, København
A. C. Meyers Vænge 15
2450 København SV, Denmark
ggusta21@student.aau.dk

Marco Timossi
Aalborg Universitet, København
A. C. Meyers Vænge 15
2450 København SV, Denmark
mtimos21@student.aau.dk

Thomas Albert Rushton
Aalborg Universitet, København
A. C. Meyers Vænge 15
2450 København SV, Denmark
trusht21@student.aau.dk

ABSTRACT

Kuplen is a fully embedded digital musical instrument, whose tactile, tiltable surface is mapped to parameters of a physical modelling synthesis sound engine. The underlying stiff string physical model, excited by a bow with movable damping element, can produce a variety of pitches and timbral qualities.

Players reported their interest in exploring the expressive capabilities of the instrument. Indeed, by combining free-form modes of interaction with established higher-level descriptor mapping strategies, Kuplen invites the user into novel sonic explorations and gestural controls over an exploratory framework of the physical model.

1. INTRODUCTION

Kuplen is a novel instrument that seeks to explore and redefine the paradigm of interface design for physical modelling synthesis. It does that by building an experimental inter-dependent framework that proposes: 1) a novel interaction with physical modelling synthesis; 2) a physical model that matches the explorative paths of the interface.

The approachable mappings and the transparent mode of interaction make it an instrument that complies with the Cook's principles of *instant music, subtlety later* [1]. However, its manifold timbral and tonal qualities and the numerous instrumental gestures associated with it highlight an underling layer of complexity. Early stage evaluation proved its potential as an instrument with *low entry fee, but no ceiling on virtuosity* [2].

2. DESIGN AND IMPLEMENTATION

2.1 Construction

With its appearance resembling that of a mushroom, the instrument consists of two distinct bodies. The top part is a $220\,\mathrm{mm}$ diameter curved *dome* constructed from laser-cut acrylic slats, into which a capacitive grid of $0.7\,\mathrm{mm}$ copper-wired rows and columns is threaded. The dome is attached to an open disk surface, the *wheel*.

Copyright: © 2022 Gabriel Vincent Karl Gustafsson et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

The spokes of the wheel are $3\,\mathrm{mm}$ elastic strings attached to the rim and terminating at a central hub, which is partially enclosed within the hollow top of the lower body. The elastic properties of the spokes allow the dome to bounce, tilt, and be rotated about its axes.

The *lower body* of the instrument, formed from stacked laser-cut $3\,\mathrm{mm}$ HDF slices of progressively increasing diameter provides space to accommodate the embedded Bela [1] hardware platform. Finally the two distinct bodies are connected by means of four additional elastic strings, which are threaded between the wheel and four corresponding eye hooks attached to the lower body. In this way, not only is the body guaranteed a degree of stability, but a level of tension is also added to the four corners of the wheel.

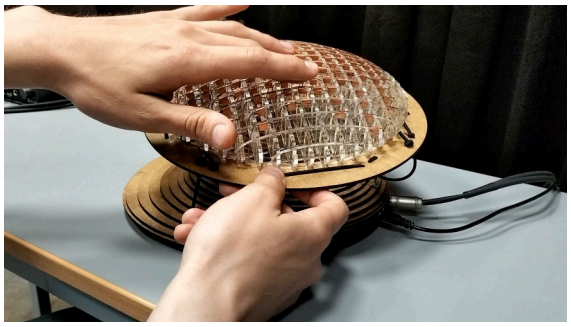


Figure 1: Kuplen in performance.

2.1.1 Sensors

The sensor system is constituted of a *Trill Craft* capacitive touch sensor and a GY-521 inertial measurement unit (IMU).

The *Trill Craft* is used to retrieve the position and pressure data from the capacitive grid. Position is derived by detecting touch at the intersection of each row and column, while pressure corresponds to the amount of force applied onto the surface. The IMU measures the tilt orientation of the upper body, expressed as pitch and roll.

The real-time stream of data is fed into Bela via the I2C protocol and used to control various parameters of a simulated bow and damping element.

2.2 Sound Engine

Bela's comparatively powerful CPU and compile-time vectorisation permitted implementation in C++ of a physical

[1] https://bela.io

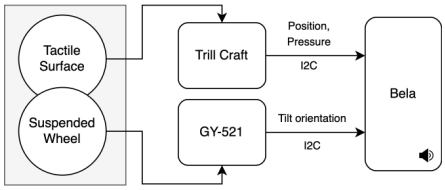


Figure 2: Sensor data flow

model based on finite difference time domain methods, as described in [3] and [4]. The following finite difference scheme forms the basis for the synthesis engine:

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot} + 2\sigma_1\delta_{t-}\delta_{xx}u_l^n - J_b(x_B^n)F_B^n\phi(v_{rel}^n) + J_p(x_P^n)F_P^n, \quad (1)$$

This equation describes the displacement u of a stiff string at position l and timestep n, with frequency-independent (σ_0) and frequency-dependent (σ_1) losses, a bowed excitation, and a massless spring-damping element. c is the wavespeed in the string, κ a combined stiffness parameter; F_B^n and F_P^n are time-varying forces applied by the bow, and the damping element respectively. $J_i(x_I)$ is an ith-order spreading operator that distributes the effects of the bow and damper at continuous string positions x_B and x_P. $v_{rel}^n = \delta_{t\cdot}u_l^n - v_B^n$, is the relative velocity between the bow and string, and ϕ is a static friction model for the bow:

$$\phi(v_{rel}^n) = \sqrt{2a}v_{rel}e^{-a(v_{rel}^n)^2+(1/2)} \quad (2)$$

Though having no explicit solution, equation (2) is continuous and differentiable, thus v_{rel}^n can be computed iteratively.

The damper is a combination of a simple linear oscillator, a cubic nonlinear oscillator, plus a localised loss term:

$$F_P^n = -\omega_0^2\mu_{t\cdot}\eta^n - \omega_1^4(\eta^n)^2\mu_{t\cdot}\eta^n - 2\sigma_P\delta_{t\cdot}\eta^n, \quad (3)$$

where $\eta = I_p(x_P^n)u_l$ is a pth order interpolator applied at the continuous damper location x_P. ω_0, ω_1, and σ_P can be adjusted independently or in combination to create different *preparation* effects at the damping position.

Output is taken as the interpolated velocity of the string at continuous position x_I:

$$y[n] = (1/2k)\left(I_j(x_I)u_l^n - I_j(x_I)u_l^{n-2}\right), \quad (4)$$

where $k = 1/f_s$ is the sampling interval. Two output positions were used, and output delivered independently to Bela's stereo output channels.

2.3 Mappings

The approach to mapping was based on Hunt's method of abstracting control and musical variables into higher level descriptors, e.g. control input = energy, synthesis parameter = brightness [5]. Such a model, inspired by the core functions of acoustic instruments, describes a straightforward and organic strategy for the cross-mapping of parameters.

Along those lines, the musical variables of the model were categorised as relating to pitch, timbre and amplitude. Mappings were devised as shown in Figure 3, following an iterative process involving parallel adjustment of the relationship between the control variables and musical criteria.

	IMU Pitch	IMU Roll	Row Position	Column Position	Row Magnitude	Column Magnitude
Pitch	Damper Position (-1, 1)	Damper Position (-1, 1)	N/A	N/A	N/A	N/A
Timbre	Damper Position (-1, 1)	Damper Position (-1, 1)	Bow Position (0, 0.5)	Damper Position (-1, 1)	Bow Velocity (-0.4, 0.4) Bow Force (0, 0.4)	Damper Non Linearity (0; 900)
Amplitude	Bow Velocity (-0.4, 0.4)	Bow Velocity (-0.4, 0.4)	N/A	N/A	Bow Velocity (-0.4, 0.4) Bow Force (0, 0.4)	N/A

Figure 3: The mappings

3. CONCLUSION & PERFORMANCE METHODS

Initial test performers reported that Kuplen provides a rewarding, exploratory musical interaction. The sensitivity of its tactile surface facilitates instant sound generation, and the mapping scheme offers subtlety of control.

Performers may touch the surface to excite the modelled string at its fundamental frequency, then slide their finger towards a different location, moving the bow and changing the timbre. Tilting the upper body can then adjust the bow velocity. Additional touches may introduce the damping element, dividing the string in two and altering the pitch. Rapid introduction and removal of the damping element may give rise to transient, 'hammer-on' effects, and percussive sounds.

In light of its affordances, *Kuplen* supports multiple playing techniques, that allow the combination of a wide array of gestures into a fluent stream of control data. The distorted, rattling high-pitched textures achieved by tilting the upper body while simultaneously tapping the surface manifest the potential of its expressive capabilities.

4. REFERENCES

[1] P. Cook, "2001: Principles for designing computer music controllers," in *A NIME Reader*. Springer, 2017, pp. 1–13.

[2] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," *Computer music journal*, vol. 26, no. 3, pp. 11–22, 2002.

[3] S. Bilbao, *Numerical sound synthesis: finite difference schemes and simulation in musical acoustics*. John Wiley & Sons, 2009.

[4] S. Willemsen, "The emulated ensemble: Real-time simulation of musical instruments using finite-difference time-domain methods," 2021.

[5] A. Hunt, M. M. Wanderley, and M. Paradis, "2002: The importance of parameter mapping in electronic instrument design," in *A NIME Reader*. Springer, 2017, pp. 29–44.

Participation in the SMC-2022 conference was supported by the European Art Science and Technology Network (EASTN-DC)

IFC-22 Papers

FEEDBACK ACOUSTIC NOISE CONTROL WITH FAUST ON FPGA : APPLICATION TO NOISE REDUCTION IN HEADPHONES

Loïc Alexandre

LMFA, UMR5509
Univ Lyon, Ecole Centrale de Lyon, CNRS, Univ Claude Bernard Lyon 1, INSA Lyon, 69130, Ecully, France
loic.alexandre@ec-lyon.fr

Pierre Lecomte

LMFA, UMR5509
Univ Lyon, INSA Lyon, CNRS, Ecole Centrale de Lyon, Univ Claude Bernard Lyon 1, 69621, Villeurbanne France
pierre.lecomte@univ-lyon1.fr

Marie-Annick Galland

LMFA, UMR5509
Univ Lyon, Ecole Centrale de Lyon, CNRS, Univ Claude Bernard Lyon 1, INSA Lyon, 69130, Ecully, France
marie-annick.galland@ec-lyon.fr

Maxime Popoff

CITI, EA3720
Univ Lyon, INSA Lyon, Inria, 69621 Villeurbanne, France
maxime.popoff@insa-lyon.fr

ABSTRACT

This work studies the feedback Active Noise Control (ANC) in a headphone with a digital filter implemented on a FPGA instead of an analog filter, in the context of a pedagogical practice bench. The digital approach allows greater flexibility in setting the feedback ANC filter but brings an additional latency that can compromise the ANC efficiency. The principle of feedback ANC in headphones is reviewed and the choice of a biquadratic filter as a feedback ANC filter is argued. The digital filter is programmed in the FAUST language and compiled on a FPGA platform. An experimental validation is carried out to compare the attenuation performance between the digital and analog biquadratic filters. The results show similar or even better low frequency attenuation for some configurations of the digital biquadratic filter compared to the analog filter. Finally, a digital filter cascading two biquads is studied and shows a bandwidth broadening where the ANC is effective.

1. INTRODUCTION

Active Noise Control (ANC) is an established solution for noise reduction in a large variety of applications [1]. In the case of headphones, ANC has become a technological and commercial asset by providing noise reduction in the low frequency band where the passive solution performs poorly. Based on the superposition of an external annoying sound with an "anti-sound", different approaches are possible to process the incoming unwanted sound and generate, in real-time, the canceling filtered sound. The main differences result from the ANC filter, referred to as the control filter throughout this paper. One of the first techniques described in the literature is the feedback approach. The sound and anti-sound signals are measured in the vicinity of the control region and fed back to the active noise control loudspeaker after passing through an ANC feedback control filter. The early implementation uses an analog filter as the control filter [2]. On one hand, the low-latency characteristics of the analog filter allows the control loop to generate the canceling sound instantly. On the other hand, the characteristics of the analog filter are fixed and the controlled frequency band cannot be adapted. Nowadays, the improved performance and the accessibility to low cost Digital Signal Processing (DSP) boards offer the opportunity to implement adaptive filtering algorithms in ANC headphone systems [3, 4]. However, with digital filtering appears the issue of latency due to Analog to Digital Converter (ADC) and Digital to Analog Converter (DAC). A solution to keep the convenience of programmable boards without obtaining unreasonable processing latency is the use of Field Programmable Gate Arrays (FPGAs) [5]. Based on configurable logic blocks, their basic architecture makes them faster than DSP boards and allows complex filtering configurations [6]. Unfortunately, the FPGAs' architecture also complicates significantly their programming.

In this context, the Fast Audio Signal-processing Technologies on FPGA [1] (FAST) project aims to simplify the FPGA programming by offering the compilation of FAUST algorithms onto FPGA boards [7]. The FAST programming tool is used in this paper in an academic context. Indeed, the authors' university proposes a training course on ANC in which an educational bench dedicated to the study of feedback ANC in headphones is used. The educational bench is shown in Fig. 1. The feedback ANC is currently operated using an analog biquadratic (biquad) filter with pre-set and fixed parameters : the box on the shelve on bottom left part of Fig. 1. The goal of this work is to replace this analog filter by a programmable digital biquad filter on FPGA with very low latency : the electronic boards on the middle of the table in Fig. 1. The FPGA circuit is synthesized from a FAUST code using the tool described in [7]. This implementation is considered more flexible as the digital filter parameters can easily be changed, which serves the educational purpose of the bench.

The outline of the paper is as follows: In Sec. 2 the principle of feedback ANC is reviewed. In particular, the specification of the feedback control filters is presented in Sec. 2.1 and the choice of a biquadratic control filter is argued in Sec. 2.2. The implementation on a FPGA board with FAUST is introduced in Sec. 3. An experimental validation of the proposed implementation is described in Sec. 4 and discussed in Sec. 5, especially in comparison with the current analog system used on the educational bench. Finally, the paper concludes in Sec 6.

[1] https://fast.grame.fr/

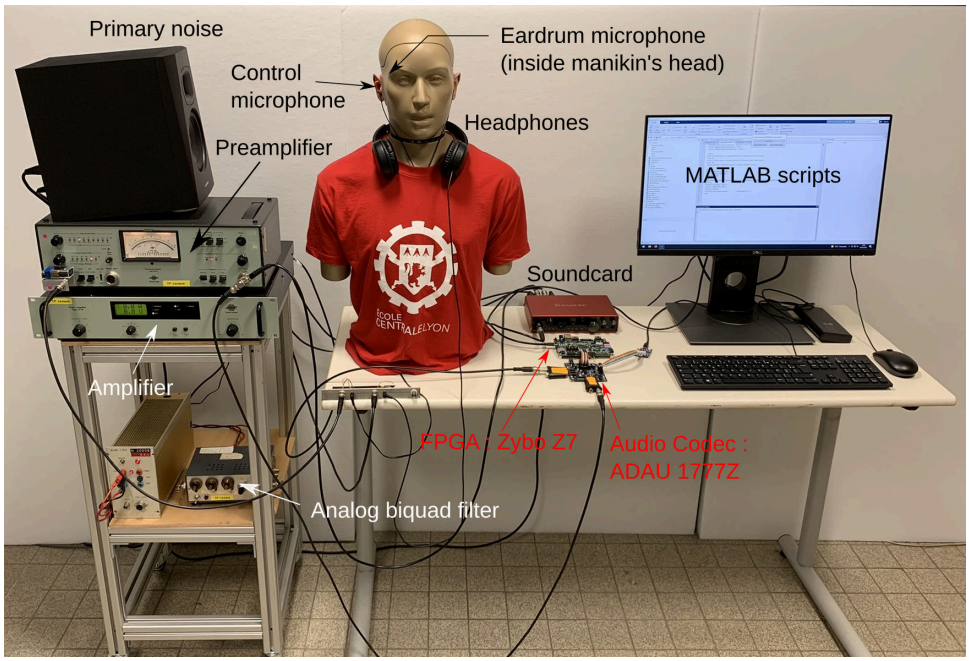


Figure 1: *Experimental set-up of the feedback ANC bench at Ecole Centrale Lyon. The set-up allows to study feedback ANC in headphones for academic purpose. The measurements are performed using ®MATLAB, an external soundcard and a loudspeaker for primary noise generation. The amplified headphones are placed on a head and torso simulator with eardrum microphone and a preamplified error microphone placed in its ear. An analog biquadratic filter is used as a control filter. In the current work, a digital biquadratic filter programmed on a FPGA board with external audio codec can be used as control filter for the feedback ANC.*

2. FEEDBACK ACTIVE NOISE CONTROL

2.1. Feedback with a Control Filter

This section describes the principle of the feedback ANC in headphones with a control filter as studied in [2]. The principle is depicted in Fig. 2 : a primary noise induces a pressure p_p at a control microphone placed inside a headset. A secondary noise is generated by the headset speaker and induces a pressure p_s at the control microphone. Therefore, the total pressure at the control microphone is given by $p_t = p_p + p_s$. The active noise control principle is to generate p_s, using a control filter H and a gain G to attenuate the total pressure p_t with respect to p_p. The control microphone is placed close enough to the listener's eardrum so that the pressure at the latter is assumed to be equal to that at the control microphone. Thus, as p_t decreases, the listener feels an attenuation of the noise from the primary source. The attenuation level and bandwidth depends on the ANC efficiency.

Attenuation The total pressure at the control microphone is given in the frequency domain by:

$$p_t(f) = p_p(f) + p_s(f) = p_p(f) + p_t(f)H(f)GB(f), \quad (1)$$

where f is the frequency in Hz, $H(f)$ is the control filter Frequency Response Function (FRF), G a frequency-independent gain and $B(f)$ the open loop FRF between the headphone speaker and the control microphone. From Eq. (1), for $p_p(f) \neq 0$, the attenuation can be written as:

$$\frac{p_t}{p_p}(f) = \frac{1}{1 - H(f)GB(f)} \quad (2)$$

Thus, in order to minimize the ratio $|\frac{p_t}{p_p}|$ the denominator in the right-hand side of Eq. (2) should be maximized. An attenuation occurs when $|H(f)GB(f)| \gg 1$. The greater the $|H(f)GB(f)|$ the greater the attenuation. As $B(f)$ is fixed from the headphone configuration, the role of $H(f)G$ is to increase the gain of the closed loop $H(f)GB(f)$ in the frequency range where attenuation is desired. However, when increasing the gain, the system stability must be considered. In fact, instability occurs with an audio feedback when $H(f)GB(f) = 1$.

Control loop with a gain G To better understand the requirements of control filter $H(f)$ and gain G for an effective attenuation, the situation where $H(f) = 1$ is first considered. A typical FRF $GB(f)$ is shown in Fig. 3. The active attenuation is desired in the low frequency range where the passive attenuation of the headphone is poor.

When one increases the gain G, the modulus response $|GB(f)|$ is shifted upwards. Therefore, from Eq. (2) the noise at frequencies such that $|H(f)GB(f)| \gg 1$ will be attenuated. However, in the reported case of Fig. 3, it can be observed that the phase response $\arg(GB(f))$ crosses $0°$ around $f_{0°} = 5169$ Hz. At this value, the transfer function is real, and if the gain G increases such that the modulus at 5169 Hz crosses 0 dB, one obtains, $|GB(f_{0°})| = 1$. The system becomes unstable. Moreover, the gain margin of 2.2 dB before instability, is not sufficient to verify the condition $|H(f)GB(f)| \gg 1$ in the low frequency region, where the atten-

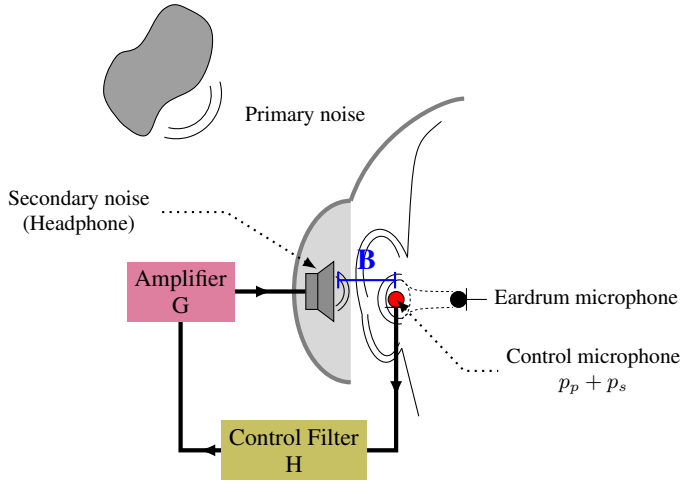


Figure 2: *Feedback ANC using a control loop with headphones: a control microphone is placed close to the eardrum in the headphone cavity. This microphone receives a pressure from a primary noise p_p and a secondary noise p_s from the headphone's speaker. With a control loop composed of a filter H and an amplifier G, the pressure level from the external primary noise is attenuated at the control microphone and at the eardrum in its vicinity.*

uation is desired. In this case one cannot have attenuation without instability. A control filter is thus introduced to achieve an efficient noise reduction system.

Control filter $H(f)$ requirements The control filter should increase the modulus of $|GB(f)|$ FRF as much as possible with respect to f_{0°. The low frequency region of the spectrum is targeted, between f_1 and f_2. Thus, the control filter $H(f)$ should be such that $\forall f \in [f_1, f_2]$:

$$|H(f)GB(f)| \gg 1 \text{ AND } \arg(H(f)GB(f)) \neq 0. \quad (3)$$

2.2. Biquadratic Transfer Function as Control Filter $H(f)$

An ideal solution for $H(f)$ to achieve Eq. (3) is $-B^{-1}(f)$. However, the latter is not causal and therefore cannot be implemented [2]. In the pedagogical bench context, a biquadratic filter [8] is used for $H(f)$ by using two resonant filters: a filter with a pair of complex conjugate zeros[2], and a filter with a pair of complex conjugate poles[3]. This choice is justified by the fact that the phase response can be set such that it does not change that of $\arg(GB(f))$ around f_{0°. The biquadratic analog transfer function is given by:

$$H(s) = \frac{s^2 + 4\pi f_z \xi_z s + (2\pi f_z)^2}{s^2 + 4\pi f_p \xi_p s + (2\pi f_p)^2}, \quad (4)$$

where s is the Laplace's variable, f_z and f_p are the resonant frequencies of the two zeros and two poles filters respectively. ξ_z and ξ_p are the damping factors of the two-zero and two-pole filters respectively.

[2] `https://ccrma.stanford.edu/~jos/fp/Two_Zero.html`

[3] `https://ccrma.stanford.edu/~jos/fp/Two_Pole.html`

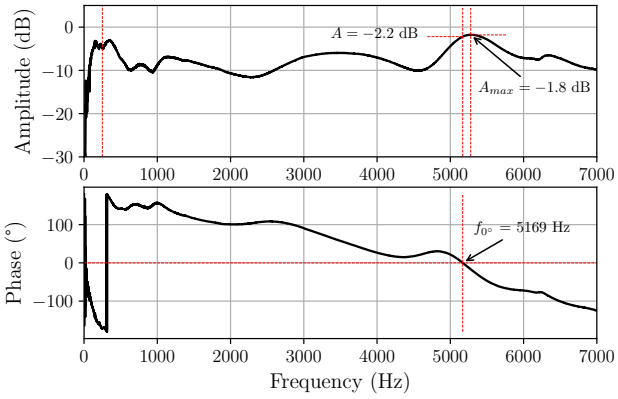


Figure 3: *Modulus (top graph) and phase (bottom graph) response of the transfer function $GB(f)$ between the amplifier and the control microphone. The FRF is measured in the experiment described in Sec. 4. The phase response crosses 0° at $f_{0^\circ} = 5169$ Hz. At this frequency the modulus is $A = -2.2$ dB. The maximum amplitude is at $f = 5278$ Hz at $A_{max} = -1.8$ dB.*

Two-pole filter The two-pole filters allow to reinforce a frequency band by adjusting the resonance frequency f_p and the damping factor ξ_p. The resonance can be set, for example, to a maximum of $|GB(f)|$ spectrum, in the low frequency region before f_{0°. For instance, from Fig. 3 one can choose $f_p = 250$ Hz. The dynamic range available for attenuation at this frequency before the instability occurs is then increased, as discussed in Sec. 2. Another strategy is to adjust the resonance frequency f_p to boost another zone of the low frequency region of $|GB(f)|$ spectrum. This will improve the attenuation at this frequency and broaden the frequency band where attenuation is possible. However, this strategy will provide less dynamic range compared to a situation where the resonance would be placed on a maximum. Thus, a compromise between bandwidth and dynamic attenuation should be made by the user when setting the parameters f_p and ξ_p.

Phase response A two-pole filter rotates the phase by -90° after the resonance frequency f_p. The transition zone between 0° and -90° depends on the value of ξ_p. Thus, if one were to use only a two-pole filter, $\arg(GB(f))$ would cross 0° at a lower frequency in the spectrum, which is not desirable as this limits the bandwidth of the possible attenuation. The use of a two-zero filter allows to overcome this problem. Indeed, the latter rotates the phase by $+90^\circ$ after the resonance frequency f_z. Thus, by placing the resonance frequency f_z close to f_p and by tuning the values of ξ_p and ξ_z, the use of the biquadratic filter does not change the value of f_{0°. This type of filter thus makes it possible to answer to a certain extent the criterion of Eq (3).

Analog and digital biquad filters Three examples of biquadratic FRF are shown in Fig. 4. The resulting FRF of the control loop $H(f)GB(f)$ are shown in Fig. 5.

The black solid line corresponds to the response of the analog biquad filter cell of the educational bench. The phase response $\arg(H_a(f))$ is almost null except in the vicinity of the resonances frequencies f_p and f_z of the poles and zeros pair respectively. In

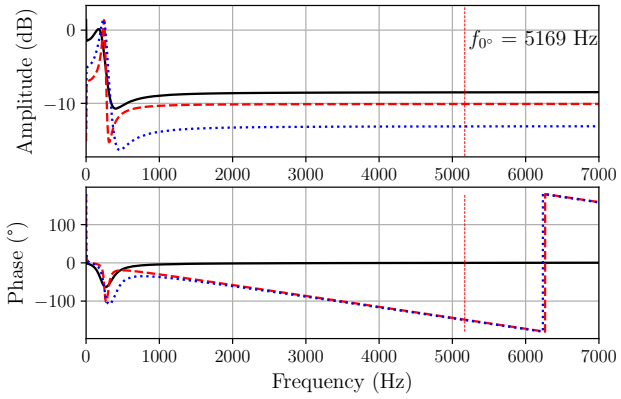


Figure 4: *Modulus (top graph) and phase (bottom graph) response of biquadratic filters transfer functions. Black solid line: $H_a(f)$ analog biquad filter of the educational bench (see Fig. 1). Red dashed line $H_{d1}(f)$ and blue dotted line $H_{d2}(f)$: two realizations of digital biquad filters on FPGA Zybo Z7 + ADAU 1777Z audio codec. The digital filters parameters for $H_{d1}(f)$ are: $f_p = 250$ Hz, $f_z = 300$ Hz, $\xi_p = 0.07$, $\xi_z = 0.1$ and for $H_{d2}(f)$: $f_p = 250$ Hz, $f_z = 400$ Hz, $\xi_p = 0.16$, $\xi_z = 0.25$. Frequency where the phase response crosses 0 (f_0) is indicated with a vertical dashed line.*

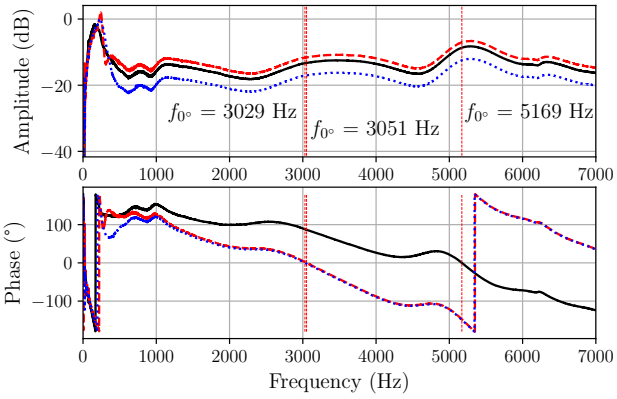


Figure 5: *Modulus (top graph) and phase (bottom graph) response of closed-loop transfer function $H(f)GB(f)$. Black solid line: $H_a(f)GB(f)$. Red dashed line $H_{d1}(f)GB(f)$ and blue dotted line $H_{d2}GB(f)$. Frequency where the phase response crosses 0 (f_0) is indicated with a vertical dashed line.*

particular, the fact that the filter is analog results in an almost null group delay, which does not modify f_{0° of $GB(f)$ and therefore does not change the ANC theoretical bandwidth.

The red dashed and blue dotted curves correspond to two digital realizations of the biquad filter with slightly different parameters. In comparison with the analog filter, one can see a slope in the phase response $\arg(H_{d1}(f))$ and $\arg(H_{d2}(f))$. This constant group delay is due to the latency the ADC and DAC of the audio codec .

Thus, the phase slope of $\arg(H_{d1,2}(f))$ impacts the phase response of $\arg(H_{d1,2}(f)GB(f))$ and decreases the value of f_{0° as it can be seen in Fig. 5. The f_{0° frequencies are respectively 3029 Hz and 3051 Hz for the digital filters presented compared to 5169 Hz for the analog filter. The ANC bandwidth is therefore reduced in this case. A theoretical analysis taking into account sine signals and a system with a flat frequency response, as in [4, Sec. 3.2], suggests that a digital filter with the lowest possible latency is desirable for feedback ANC. However, while the ANC bandwidth narrows as the latency increases, the achievable attenuation in the given frequency bandwidth is also highly dependent on the system frequency response. In the case of the FRF $GB(f)$ considered here, the shift of the unstable frequency due to the latency leads to a higher dynamic for attenuation in the low frequency region. Indeed, the amplitude of $GB(f)$ being lower at the shifted unstable frequency, the amplitude margin before instability is greater. This allows to further increase the gain G and therefore improves the attenuation. The topic of latency effect on ANC performance is left for further investigations.

In the three cases, the control filter $B(f)$ allows to raise the low frequency region of the spectrum $H(f)GB(f)$ below the instability and improves the attenuation dynamic according to Eqs. (2) and (3). It should be kept in mind that the digital filters can be adjusted online and that the passive attenuation of the headphones also helps in the high frequency region.

3. IMPLEMENTATION WITH FAUST ON FPGA

As discussed in Sec. 2.2, the control filter $H(f)$ for this work is a biquad filter. Its implementation on the FPGA is done using the workflow proposed in [7]. The FPGA evaluation board Zybo Z7 is used for this work alongside an audio codec ADAU1777Z. The latter has a round trip latency around 80 μs at a sampling frequency of 192 KHz.

Single biquad implementation The biquad transfer function is directly taken from Eq. (4) in the analog domain and transformed into a digital filter by a bilinear transform [9, Sec. 7.2]. This operation is performed using the function `tf2s` from the library `filters.lib` from the FAUST distribution. The resulting code is given hereafter:

```
import("stdfaust.lib");

fp = hslider("Freq Pole",150,50,1000,0.1);
fz = hslider("Freq Zero",150,50,1000,0.1);
xip = hslider("Tau Pole",0.1,0,1,0.01);
xiz = hslider("Tau Zero",0.1,0,1,0.01);

b1 = 4 * ma.PI * fz * xiz;
b0 = (2 * ma.PI * fz)^2;

a1 = 4 * ma.PI * fp * xip;
a0 = (2 * ma.PI * fp)^2;

w1 = 1;
// w1 is the desired digital frequency
//(in radians/second) corresponding to the
//analog frequency 1 rad/sec (i.e., s = j).

process = _ : fi.tf2s(1,b1,b0,a1,a0,w1);
```

Note that the user can control at execution time the resonant frequency of the pole f_p and the zero f_z, as well as their respective damping rates ξ_p and ξ_z using a user interface generated on the computer or with a hardware board as presented in [7].

Cascaded biquad implementation In order to improve the ANC bandwidth as discussed in Sec. 2, another version of the $H(f)$ control filter is implemented as a cascade of two biquads. In this case, the user can control the parameters of each of the biquad sections of the filter and activate or bypass them using a checkbox. The FAUST code is the following:

```
import("stdfaust.lib");

fp(i) = hslider("[1]Freq Pole %i
    ",150,50,1000,0.1);
fz(i) = hslider("[3]Freq Zero %i
    ",150,50,1000,0.1);
xip(i) = hslider("[2]Tau Pole %i
    ",0.1,0,1,0.01);
xiz(i) = hslider("[4]Tau Zero %i
    ",0.1,0,1,0.01);

off(i) = 1-checkbox("[5]Off %i");

b1(i) = 4 * ma.PI * fz(i) * tz(i) * off(i
    );
b0(i) = (2 * ma.PI * fz(i))^2 * off(i);

a1(i) = 4 * ma.PI * fp(i) * tp(i) * off(i
    );
a0(i) = (2 * ma.PI * fp(i))^2 * off(i);

w1 = 1;

process = _ : seq(i, 2, fi.tf2s(1,b1(i),
    b0(i),a1(i),a0(i),w1));
```

4. EXPERIMENTAL VALIDATION

Experimental layout The experimental layout is visible in Fig. 1. An instrumented dummy head (®G.R.A.S Kemar 45BB head and torso simulator) is used. Only the right-ear is used for the practical. The feedback control loop ($H(f)GB(f)$) is composed of a ®JVC HA-S31M-B headphones for which the input signal is amplified by a ®B&K Type 2718 power amplifier (G). The control microphone inside the loop is a standard electret microphone placed in a 3D printed support [4]. Its output signal is preamplified by a ®B&K Type 2610 preamplifier with fixed gain and goes to the feedback loop biquad filter which is either:

- An analog biquad filter, $H_a(f)$, as visible on the lower shelf on the bottom left side of Fig. 1,
- A digital biquad filter, $H_d(f)$ implemented on a ®Xilinx ZYBO Z7 FPGA evaluation board alongside an ADAU 1777Z audio codec.

The primary noise source uses a low-pass filtered white noise played on a ®Adam Audio T5V loudspeaker.

[4] http://sekisushai.net/ambitools/2020/12/02/make-your-own-binaural-microphones.html

Acquisition and primary noise signal generation are performed by a ®Focusrite Scarlett 8i6 3rd Gen soundcard. FRF measurement and spectrum displays are done with ®MATLAB scripts.

The sound pressure level of the noise produced by the external loudspeaker is about 70.6 dBA at the exterior of the manikin ear.

Protocol The experimental protocol consists in measuring first the FRF $GB(f)$. Its analysis allows to identify the frequency band for which ANC is possible, in particular to determine f_{0° as described in Sec.2. Then, the biquadratic control filter $H(f)$ is designed in adequacy. Although many solutions exist to choose the poles' and zeros' characteristics of the control filter [6], the method adopted here is empirical with the intention of observing the impact of the biquad parameters on the sound pressure level attenuation. Therefore, the filter parameters f_p, f_z, ξ_p and ξ_z are set manually from the analysis of FRF $GB(f)$. Once $H(f)$ parameters are set, the transfer function $H(f)GB(f)$ is deduced and can be also measured to predict the ANC performance. Finally, the ANC performance is measured as follows: A primary noise is played in the frequency band [50-1000] Hz through the external loudspeaker and the gain G is increased until the audio feedback is observed. Finally, with the gain set to its maximum value, the measurement of the eardrum microphone signal is carried out with and without the headphones feedback ANC system. The difference between the two measurements gives the attenuation as a function of the frequency performed by the ANC feedback system.

5. RESULTS AND DISCUSSION

This section presents the results obtained for several control filter configurations. The reference configuration is the analog biquad filter $H_a(f)$ visible as solid line in Fig. 4 and its corresponding closed loop transfer function in Fig. 5.

5.1. Single Biquad Configuration

The FRF $H_{d1}(f)$ of the first digital control filter and the corresponding configuration are presented in Figs. 4 and 5 with a red dashed line. As explained in Sec. 2.2, the digital filter latency reduces $f_{0^\circ} = 5169$ Hz to $f_{0^\circ} = 3051$ Hz in comparison with the analog filter $H_a(f)$. However, as the primary noise is in the range $[50 - 1000]$ Hz, this bandwidth limitation is not relevant here and the filter $H_{d1}(f)$ can be used for feedback ANC.

One of the main advantage of the digital control filter implementation is the configuration flexibility : as explained in Sec. 3, a user interface allows modifying the filter parameters in real-time. In this matter, a second digital control filter configuration, $H_{d2}(f)$ is presented with a blue dotted line in Figs. 4 and 5.

The ANC attenuation is shown in Fig. 6 using either the $H_a(f)$ (black solid curve), $H_{d1}(f)$ (red dashed curve) or H_{d2} (blue dotted curve) as control filter. It corresponds to the ratio of pressure measured at the eardrum microphone without ANC and with ANC. It can be observed that the higher attenuation is concentrated in the frequency range [50-400] Hz, which is in accordance to the control filter configurations. It is consistent that with a single biquad filter, the frequency range operation of the ANC is limited. In the case of the analog control filter $H_a(f)$ the attenuation obtained is above 10 dB in a narrow frequency band from 154 Hz to 195 Hz. For the first digital control filter $H_{d1}(f)$, the attenuation reaches 10 dB with a maximum at 15.4 dB in the frequency region [158-261] Hz. One may observe the important peak

of attenuation due to the low damping poles and zeros coefficients chosen for this configuration. Finally, the third filter $H_{d2}(f)$ gives the most interesting results with an attenuation above 10 dB over the frequency band [102-297] Hz and with a maximum attenuation reaching 19.4 dB. This comparison of the analog filter configuration with two different digital filter configuration does not only aim to validate the employment of digital control filters but also to demonstrate the importance of configuration flexibility. As a matter of fact, if for a single biquad filter the chosen configuration can severely change the obtained noise attenuation, the design must be wisely chosen for a more complex filter.

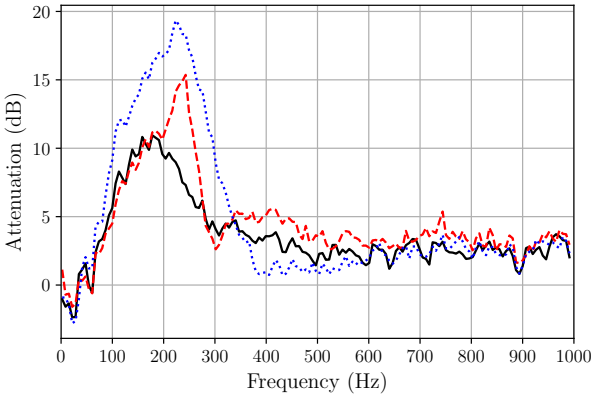


Figure 6: *Noise attenuation obtained with three different control filters $H(f)$. Black solid line: H_a. Red dashed line H_{d1} and blue dotted line H_{d2}.*

5.2. Cascade Configuration of Biquad Filters

As an extension to the reference single analog control filter, this section presents a configuration with two cascade digital biquad filters whose implementation is presented in Sec. 3.

Two configurations, $H_{d3}(f)$ and $H_{d4}(f)$ are evaluated. Their frequency response and resulting closed loop FRF $H(f)GB(f)$ are shown in Figs 7 and 8 respectively. As for the previous case with a single biquad filter, the two cascade configurations lower the frequency of instability of the closed loop. In comparison with the analog filter $H_a(f)$, $f_{0^\circ} = 5169$ Hz is shifted to $f_{0^\circ} = 2886$ Hz with $H_{d3}(f)$ and to $f_{0^\circ} = 2813$ Hz with $H_{d4}(f)$. Although the resulting frequencies of instability are much lower than the initial one, noise attenuation is still achievable in the bandwith of interest [50-1000] Hz. The setting differences between both configurations relies exclusively on the damping coefficients of the complex conjugate poles and zeros. In fact, the cascade of biquad filters $H_{d4}(f)$ is smoother, which means that the damping coefficients are higher, than the first cascade $H_{d3}(f)$. Because of the more complex filter configuration the effective frequency range of attenuation is wider than for the previous case using only a single biquad filter. The results of noise attenuation are presented in Fig. 9. In the case of the first cascade biquad filters $H_{d3}(f)$ (black solid line) the noise attenuation obtained is equal or superior to 10 dB in the frequency band [80-498] Hz with a maximum attenuation at 22 dB. Addtionally, the attenuation reaches 15 dB or more over a quite large frequency region, from 101 Hz to 466 Hz. Concerning the second cascade biquad filters $H_{d4}(f)$ (red dashed line), the attenuation reaches 10 dB and above in the bandwidth [82-546] Hz. For the threshold of 15 dB, an attenuation is obtained in the frequency band [135-483] Hz and the maximum attenuation is 21.3 dB. One can notice that the shape of the attenuation is actually similar to the shape of the corresponding cascade of biquad filters. This behavior is in line with the aim to design a control filter with amplitude resonances in the bandwith requiring noise attenuation. The design of the two biquad filters in $H_{d3}(f)$ helps to obtain these two peaks of attenuation around 160 Hz and 415 Hz. Therefore, one must modify the design of the control filter depending on the desired outcome. A further investigation on the addition of an optimization method to design the control filter is a perspective to this study.

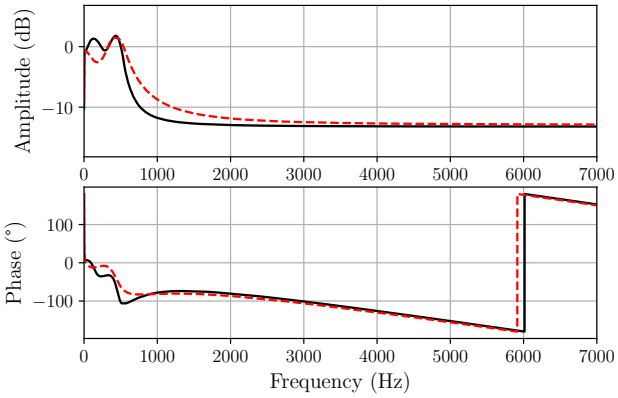


Figure 7: *Modulus (top graph) and phase (bottom graph) FRF of biquadratic filters. Black solid line: $H_{d3}(f)$ the control filter composed of two cascaded biquads with parameters : $f_{p1} = 150$ Hz, $f_{z1} = 250$ Hz, $\xi_{p1} = 0.5$, $\xi_{z1} = 1$, $f_{p2} = 450$ Hz, $f_{z2} = 550$ Hz, $\xi_{p2} = 0.125$, $\xi_{z2} = 0.5$. Red dashed line: $H_{d4}(f)$ the control filter composed of two cascaded biquads with parameters : $f_{p1} = 150$ Hz, $f_{z1} = 250$ Hz, $\xi_{p1} = 1$, $\xi_{z1} = 1$, $f_{p2} = 450$ Hz, $f_{z2} = 550$ Hz, $\xi_{p2} = 0.25$, $\xi_{z2} = 1$.*

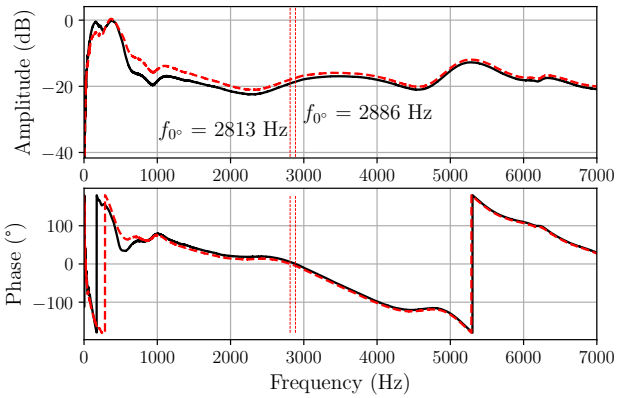


Figure 8: *Modulus (top graph) and phase (bottom graph) response of closed-loop transfer function $H(f)GB(f)$. Black solid line: $H_{d3}(f)GB(f)$ and red dashed line $H_{d4}(f)GB(f)$.*

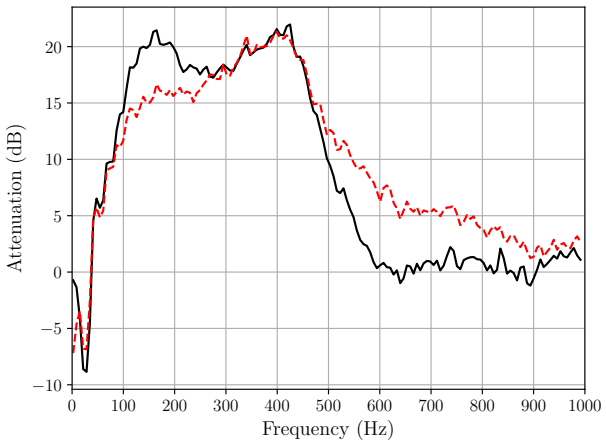


Figure 9: *Noise attenuation obtained with two different control filters $H(f)$. Black solid line: $H_{d3}(f)$ and red dashed line $H_{d4}(f)$.*

6. CONCLUSION

In this paper, the comparison between an analog and digital filters for feedback ANC in headphones was studied in the context of a pedagogical bench. The principle of feedback ANC in headphones was reviewed and the choice of a biquadratic filter as a control filter was argued. Then, the process to implement the desired control filter with FAUST was explained both for single and cascade biquad filter configurations. Furthermore, an experimental approach was proposed to validate the use of a digital control filter through the evaluation of the noise attenuation performance. Therefore, the results were presented and discussed for two biquad filter designs per configuration. In particular, a single digital biquad configuration achieved an attenuation above 10 dB in the bandwidth [102-297] Hz while the attenuation achieved is superior to 15 dB, in the frequency range [101-466] Hz, using a cascade of two biquad filters. This work is based on an existing educational bench dedicated to feedback ANC headphones. While this experiment is initially using an analog biquad as control filter, the latter was successfully replaced with a digital filter programmed on a FPGA evaluation board. The aim is to offer a wider flexibility of parameterization for pedagogical aspects. The programming workflow using the FAUST programming language greatly simplify the digital filter implementation on FPGA. A systematic way of studying, modeling and realizing feedback active control systems has been outlined here. Future works include the study of latency effect on ANC performance as well as digital filter optimization with respect to the headphone to control microphone FRF.

7. ACKNOWLEDGMENTS

This work was supported by the French National Research Agency, ANR FAST (ANR-20-CE38-0001).

This work was carried out within the LABEX CeLyA (ANR-10-LABX-0060) of the University of Lyon.

8. REFERENCES

[1] Stephen J Elliott, *Signal Processing For Active Control*, Academic Press, London, 2001.

[2] C. Carme, "Absorption acoustique active dans les cavités auditives," *Acta Acustica united with Acustica*, vol. 66, no. 5, pp. 233–246, Oct. 1988.

[3] W.S. Gan, S. Mitra, and S.M. Kuo, "Adaptive feedback active noise control headset: implementation, evaluation and its extensions," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 3, pp. 975–982, Aug. 2005, Conference Name: IEEE Transactions on Consumer Electronics.

[4] Stefan Liebich, Johannes Fabry, Peter Jax, and Peter Vary, "Signal processing challenges for active noise cancellation headphones," in *Speech Communication; 13th ITG-Symposium*. 2018, pp. 1–5, VDE.

[5] Piero Rivera Benois, Patrick Nowak, and Udo Zölzer, "Fully digital implementaiton of a hybrid feedback structure for broadband active noise control in headphones," 07 2017.

[6] Jiajie Wang, Jinhui Zhang, Jian Xu, Chengshi Zheng, and Xiaodong Li, "An optimization framework for designing robust cascade biquad feedback controllers on active noise cancellation headphones," *Applied Acoustics*, vol. 179, pp. 108081, Aug. 2021.

[7] Maxime Popoff, Romain Michon, Tanguy Risset, Yann Orlarey, and Stephane Letz, "Towards an FPGA-Based Compilation Flow for Ultra-Low Latency Audio Signal Processing," in *SMC*, Saint Étienne, 2022.

[8] Behzad Razavi, "The biquadratic filter [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 10, no. 2, pp. 11–109, 2018.

[9] Alan V Oppenheim and Ronald W Schafer, *Discrete-Time Signal Processing*, Pearson Higher Education, Upper Saddle River, 3rd edition, 2010.

AUTOMATICALLY GENERATING EURORACK HARDWARE RUNNING FAUST PROGRAMS USING EURORACK-BLOCKS

Raphaël Dingé

Ohm Force, 8 le Nézert 22340 Trébrivan, France
raphael.dinge@ohmforce.com

Stéphane Letz

Univ Lyon, GRAME-CNCM, INSA Lyon, Inria, CITI, EA3720,
69621 Villeurbanne, France
letz@grame.fr

ABSTRACT

This paper describes the integration of FAUST with Eurorack-blocks, a framework capable of generating programmatically Eurorack digital modules' hardware and firmware files for manufacturing, and providing a virtual environment for early-stage design, development, testing and debugging iterations on a desktop computer.

It presents a method to statically bind the inherently nested structure of a FAUST DSP program with the flat namespace and different types of the ERBUI and ERBB languages, which are Domain Specific Languages to describe the Eurorack module UI, module DSP file and associated audio data, respectively.

An implementation is demonstrated, taking into consideration the specific memory model of the hardware embedded platform, as well as the meta-programming technique used to minimize computations done at run time by relocating them at build time.

1. INTRODUCTION

1.1. Background

Eurorack is a modular synthesizer format originally specified in 1996 by Doepfer Musikelektronik[1]. It became since 2021 the dominant hardware modular synthesizer format, with over 12,000 modules available from more than 500 different manufacturers and individuals[2][3].

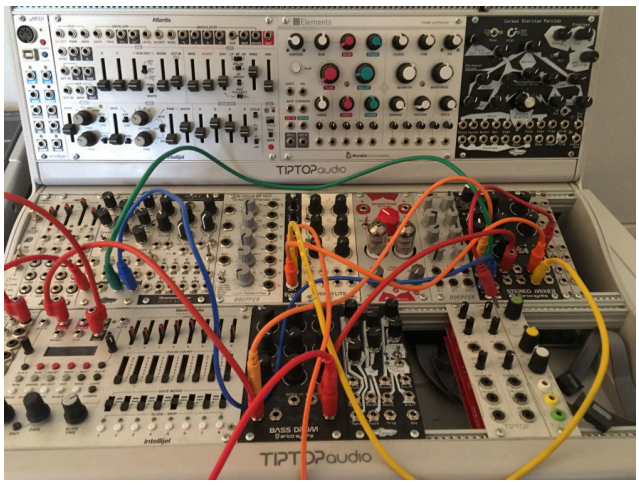


Figure 1: *A Eurorack system.*

A Eurorack system (see Figure 1) is composed of Eurorack modules that implement electrical and mechanical specifications from Doepfer, for them to properly interact with each other.

A module has a user interface that typically includes buttons, switches, pots or LEDs the usage of which is totally up to the module developer. Conversely, modules also include 3.5mm mono jack connectors used as inputs or outputs, which allow transporting a signal from one module to another, and which shall follow Doepfer electrical specifications [4].

There are 3 main types of signals: Audio, Control Voltage (CV), or Gate signals:

- An audio signal represents a mono analog audio signal, typically ranging from -5V to 5V,
- A Control Voltage represents a continuous parameter typically used for modulation, They can represent LFOs, ADSR envelopes, or pitch information, over various electrically specified ranges of voltages,
- A Gate represents a boolean parameter typically used for triggers or tempo clocks.

Eurorack users usually credit the platform for its haptic feedback and immediacy that prevents from interrupting their creative flow. Conversely though, the rapid complexity of a patch can quickly lead to errors, which can produce unexpected pleasing sounds, which users call "happy mistakes".

While the Eurorack platform was only using analog components when first introduced, more and more manufacturers are producing today digital modules which use Analog to Digital Converters (ADC) and Digital to Analog Converters (DAC) to stay compatible with the original electrical specifications, while allowing the use of embedded digital processors to produce the desired effect.

We can roughly estimate that the proportion of digital modules is approximately less than 10% of the total number of Eurorack modules produced[5]. This can be explained by the quite complex setup needed to just start any kind of work, while most individuals can start to develop analog modules with a few low-cost electronic components. Another limiting factor is the power consumption of digital modules compared to their analog counterparts, which restricts the models of processors usable in practice, as the maximum total consumption for a Eurorack system is usually around 40W.

[1] https://doepfer.de/a100e.htm
[2] https://www.modulargrid.net/e/vendors
[3] https://www.modulargrid.net/e/modules/browser

[4] In practice manufacturers bend some rules, but in a way that came to a general consensus. For example, LFOs are using usually audio signal voltage ranges instead of the specified half the range, and Gate signals can go up to 10V instead of the specified 5V

[5] https://www.modulargrid.net/e/modules/browser?SearchFunction=12

Based on this observation, the company Electro-smith [6] started to develop the Daisy Seed board which combines on a single board, a low-consumption ARM embedded processor, SDRAM memory as well as a stereo audio codec, and they developed a software library which abstracts low-level embedded software development for this hardware board.

The project was brought to a crowd-funding platform and funding was quite a success. Electro-smith produced the Daisy Patch Eurorack module (see Figure 2), which brings a predefined user interface and uses the Daisy Seed board while adapting signals to the Eurorack electrical specifications.

Figure 2: *The Daisy Patch combines a predefined set of 4 knobs and linked CV inputs, 4 audio inputs and outputs, 2 CV outputs, 2 Gate inputs, 1 Gate output, 1 MIDI input and output, a rotary encoder, OLED screen and SD card reader. The Daisy Seed board can be seen on the lower right of the picture.*

At the beginning of 2022, Electro-smith released a new Daisy Patch Submodule board [7] (see Figure 3) which was designed exclusively with Eurorack in mind, by implementing Eurorack electrical specifications directly onto the board, to further ease the development of Eurorack modules for manufacturers and individuals.

Using the Daisy Patch Submodule board still requires knowledge in hardware engineering, and compliance with the mechanical specifications dictated by Doepfer.

Our open-source project, Eurorack-blocks [8], focuses on this part, since the number of engineers who have both strong hardware and software expertise is quite low.

The Eurorack-blocks allows a software developer to develop their own custom Eurorack module, with a custom user-defined interface, by allowing them to design, develop, test, debug and iterate on their desktop computer with their development platform of choice, using a simulator and a virtual Eurorack environment, before they even start to produce the hardware.

This allows for a much more Agile approach to Eurorack development, which makes Eurorack-blocks at least a good option for prototyping, as producing a prototype requires fewer team interactions in a company, as a software developer can produce their

[6] `https://www.electro-smith.com`
[7] `https://www.electro-smith.com/daisy/patch-sm`
[8] `https://github.com/ohmtech-rdi/eurorack-blocks`

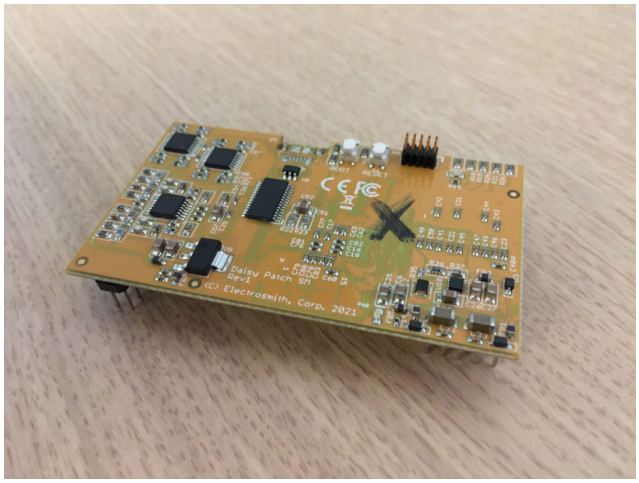


Figure 3: *The Daisy Patch Submodule board.*

prototype in isolation, without requiring the help of a hardware engineer colleague.

The simulator is a thin abstraction layer between the user code and the target platform, and the latter can be either the virtual Eurorack environment or the Daisy Patch Submodule hardware board. It is done in a way that a program that can build for the virtual environment will also build for the real environment. The simulator can also detect problems that the firmware would have at run time, when still developing on the desktop computer.

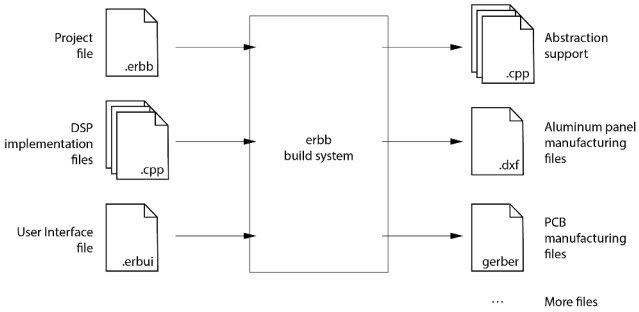


Figure 4: *The* ERBB *build system.*

On the software side, the main components of Eurorack-blocks are (see Figure 4):

- A build-system called `erbb` and its associated ERBB language, which abstracts the target platform and produces makefiles or Integrated Development Environment (IDE) project files, to either build the software for the simulator or the firmware,
- A user interface system and its associated ERBUI language, which allows describing a custom hardware user panel,
- A set of generators that produce Printed Circuit Board (PCB) production files, instruction files for front aluminum panel milling and drilling machines, PDF files for front panel silkscreen printing from the ERBUI source code, and C++ files to provide language bindings between ERBB, ERBUI and C++.

The simulator is written in C++ while the build-system is written in Python. Developers can use C++ to write their DSP code, but also `gen~` from the Max/MSP visual environment[9], or the FAUST language.

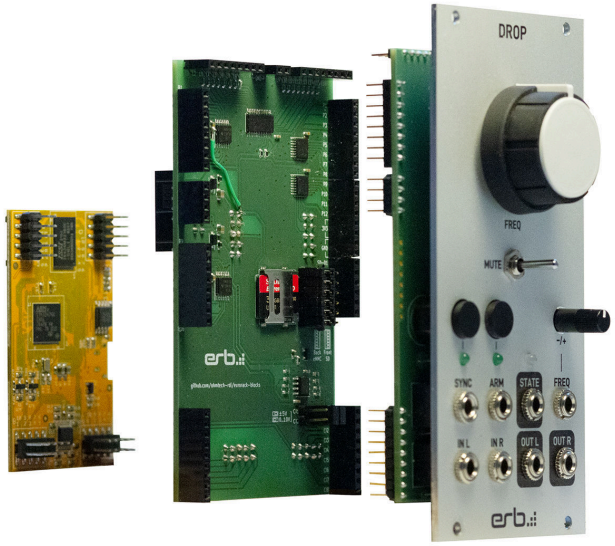


Figure 5: *From left to right, the Daisy Patch Submodule board, the Eurorack-block board "Kivu12", the front PCB tightened to the front aluminum panel with its jack connectors and pot.*

On the hardware side, a Eurorack-block project is composed of three parts (see Figure 5):

- The Daisy Patch Submodule, which provides the computation and memory units, as well as a stereo audio codec, and all the circuitry needed to adapt to the Eurorack electrical specifications,
- A Eurorack-block board, which mediates between the Daisy Patch Submodule and the front panel user-defined user interface, by extending the capabilities of the former by using multiplexers and demultiplexers. It also provides power conditioning. It is designed for mass production,
- The front PCB and its aluminum panel, which hosts all the user-defined controls of the user interface.

1.2. Simulated Environment

Making hardware can be long for different reasons, one of them being the time it takes between the moment where a design file is ready to be produced, and the time where it can really be used. Producing all the different needed parts to assemble a module, such as the Printed Circuit Board or front aluminum panel, typically takes a bit more than 6 working days to be produced and received, after which the module still needs to be assembled.

This means that the feedback loop between a change and the effectiveness of that change is around 2 weeks. In contrast, the feedback loop to which software developers are used to usually ranges from a couple of seconds to some minutes. This only in itself explains why so many companies have moved from traditional hardware making to one in which the hardware could be changed quickly using a "firmware", which drastically reduces the time to market.

This process has been enhanced even more for platforms like iOS developed by Apple, where a software developer can test their application on a specific iPhone, without even owning it. Rather, a simulated environment is used, in which a Low-Fidelity (in terms of user interactions) prototype of the application can be developed on a desktop computer.

Eurorack-blocks uses the same method. It uses VCV Rack [10], a virtual Eurorack system that can run on a desktop computer running on macOS, Linux, or Windows. We made a small layer, called the *simulator*, which tries to emulate as much as it can the real hardware. This prototype is also Low-Fidelity but is enough for most uses. Apart from processing power, it ensures that a program that can run in the simulator will also run on the real Daisy hardware (see Figure 6).

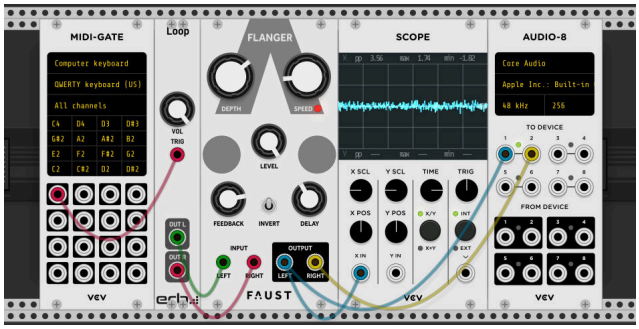


Figure 6: *The* FAUST *Flanger module running in VCV Rack. It is possible to input Audio in the Flanger, listen on the desktop connected sound interface, as well as monitor signal properties using an oscilloscope.*

Using VCV Rack is very interesting, as it offers a rich variety of modules that can be used for testing, such as LFOs, VCOs, oscilloscopes, and frequency spectrum.

The final generated module, once manufactured, can be used in a real Eurorack system (see Figure 7).

1.3. Using the ERBB DSL

ERBB is a Eurorack-blocks Domain Specific Language, which aims to simplify building a module for both the simulator and the firmware, which have very different configurations. It is called a "meta-build-system" in the sense that it is not a build-system in itself like `make`, but generates build-system files for a variety of build systems, including `make`, but also Integrated Development Environments, for different targets (here either the simulator or the firmware).

[9] `https://docs.cycling74.com/max8/vignettes/gen_topic`

[10] `https://vcvrack.com`

Figure 7: *The finalized* FAUST *Flanger module running in a Eurorack system*

For example, the following simple file for FAUST:

```
// Flanger.erbb

use strict

module Flanger {
    sources {
        file "Flanger.erbui"
        file "Flanger.dsp"
    }
}
```

Will generate:

- A `Makefile` for `make` to build a VCV Rack module to allow testing of the module in a virtual environment,
- A `Makefile` for `make` to build the Daisy firmware that will be uploaded on the hardware,
- A Xcode project for the native macOS development environment, which builds a VCV Rack module and allows to debug it,
- A VS code workspace for Linux, macOS, and Windows, which can build a VCV Rack module, the firmware, and allows to debug the firmware on the target Daisy hardware.

Additionally, it will also generate "actions" for those build systems, which:

- Detect when an ERBUI file changed to re-generate everything needed that is used during the build process,
- Generate a specific representation of audio files that can be directly compiled *into* the program,
- Order the FAUST transpiler to generate updated C++ files when the source `dsp` file changed,
- And many other task management actions to automate common actions, such as copying the built VCV Rack module to the right place, so that developers can start to test the module as soon as the build process is finished.

1.4. Using the ERBUI DSL

ERBUI is a Eurorack-blocks Domain Specific Language, which aims to simplify the physical layout of the user interface of a module. It provides standard electronic components such as potentiometers, switches, LEDs, or the standard Eurorack-compatible jack 3.5mm mono connectors.

For example, the following simple file for FAUST:

```
module Flanger {
    width 12hp
    board kivu12
    material aluminum
    header {}
    footer {}

    image "panel.svg"

    control depth Pot {
        faust { bind {
            address "/FLANGER/0x00/Depth"
        }}
        position 13.5mm, 25.645mm
        style rogan.5ps
        pin P4
    }

    control speed Pot {
        faust { bind {
            address "/FLANGER/0x00/Speed"
        }}
        position 47.1mm, 25.645mm
        style rogan.5ps
        pin P5
    }
    ...
}
```

Will generate:

- The front aluminum panel DXF file, which gives information to milling and drilling machines to produce the front panel,
- The printed circuit board (PCB) "gerber" files, which gives information for manufacturer on how to produce the board, on which controls like potentiometers, switches LEDs, or the standard Eurorack-compatible jack 3.5mm mono connector will be soldered too,
- The bill of materials (BOM), to simplify the ordering of electronic components to global component suppliers,
- A PDF file that is used to digitally print on the front aluminum panel,
- A SVG file that is used for the VCV Rack module's silkscreen,
- The C++ code to generate the module UI for the VCV Rack SDK,
- The C++ code compatible with the firmware.

1.5. Using the Faust DSL

FAUST [1] as a Domain Specific Language, specifically designed for real-time signal processing and synthesis, is a good candidate to speed up and facilitate the development of the DSP part of a Eurorack-block module.

A FAUST program describes a signal processor taking a set of audio inputs and control signals (such as buttons or sliders) and generating a set of output audio signals and controls (such as bargraphs).

After the compilation step, the controls form a hierarchy of groups nested within each other as a tree, with the terminal leaves as sliders, buttons or bargraph items. The tree description is then generated in the `buildUserInterface(...)` method as a sequence of calls to construction methods (like `openHorizontalBox/closeBox` and `addButton/addVerticalSlider`) to be implemented by an external UI object.

This object can typically interpret the tree course to build a unique path (as `/root/group1/group2/.../item` kind of string) for each tree leaf [11], or create a JSON file to describe the same information.

From the DSP source program, the FAUST transpiler will be driven by the ERBB DSL to produce a C++ class, to be compiled with specific architectures files also generated by ERBB.

2. PROBLEM STATEMENT

On one hand, the DSP files of FAUST are programs that mix the presentation (e.g. primitives like `hslider`) and the computation. A program can import libraries, which contains functions, and those functions can possibly use presentation primitives as well, and in particular layout information (e.g. primitives like `hgroup`) therefore representing a directed tree. Those primitives have a label and represent a scalar value. Two labels can have the same name in a program, as long as they are in different DSP files.

The same directed tree and labels principles apply to audio samples through the use of the `soundfile` primitive, which supports only a non-interleaved audio representation at the time of writing this paper. The `soundfile` primitive supports however the concept of "parts", which conveniently join multiple distinct audio files, into one dynamic continuous view from the DSP program perspective.

On the other hand, the ERBUI file represents a flat namespace of controls, with each control having a name and a type. A type can represent a scalar value, but also for example a boolean value, a compound type (e.g. a 2-dimensional vector for a joystick control), or possibly an entire filesystem. The computation and presentation are separated and bound at build time.

The ERBB files represent the sources (e.g. DSP file) and resources (audio samples) again in a flat namespace. Interleaved and planar audio formats are supported, but the current model only supports a one-to-one mapping between one audio file and its representation in the program.

Finally, the total number of UI primitives of a program might not allow them to fit practically on a Eurorack panel, because of space considerations, or to limit the number of features to enhance the user experience. In that case, some FAUST UI primitive will be left unbounded on purpose, and in some cases, a user might need to set a default value that might not match the one originally defined in the program (specifically in the case of imported libraries).

2.1. Constraints

Eurorack-blocks uses the Daisy Patch Submodule hardware for computation. It is an embedded platform that, while being extremely powerful for embedded use, is also very slow compared to a desktop platform, and has a different memory model than a traditional desktop computer.

Three types of different memory on the embedded platform can be used for computation, with different sizes and speeds, the larger meaning being also the slower.

Because memory and computation are very precious resources, we shall avoid computing at run time what could be computed at build time.

Finally, while the approaches exposed in the problem statement are radically different, the implementation must unify them, without requiring the user to make any modification to their past programs or the standard library of DSP functions delivered with FAUST.

2.2. Previous Work

Embedded platforms for audio processing (as microcontrollers or specialized Linux systems running on more standard CPUs like ARM) have appeared in recent years. When they run a complete OS, they can usually be programmed using C/C++ and possibly by deploying standard audio environments like PureData, Csound and SuperCollider. Some of them only contain a minimal OS, and the audio processing code has to be directly developed in C/C++ and deployed on various kinds of *bare-metal* approaches (like having the audio code directly running in the audio interrupt for instance).

Most of those platforms can directly be used with FAUST when the appropriate `faust2xx` script exists, using a set of adapted architecture files. Here are some examples of recently developed tools:

- `faust2esp32` [2], a tool to generate digital signal processing engines for the ESP32 microcontroller family. It can target both the C++ and the Arduino ESP32 programming environment and it supports a wide range of audio codecs, making it compatible with most ESP32-based prototyping boards
- `faust2teensy` [3] is a command-line application that can be used both to generate new objects for the Teensy Audio Library and standalone Teensy programs.
- `faust2bela` [12] allows to build and run a Bela project, possibly using polyphonic mode and a remote GUI.

A `faust2daisy` [13] implementation is already targeting the Daisy platform. However, it is meant for existing ready-to-use pieces of hardware, such as the Daisy Pod, while we intend to provide a development platform for a piece of hardware that doesn't already exist.

Furthermore, the virtual environment mentioned earlier is VCV Rack, a virtual Eurorack system for desktop platforms. We built an abstraction that ensures that the same code will both run on the simulator (a thin abstraction layer to the VCV Rack module SDK in charge of emulating specific behaviors of the embedded hardware) and on the embedded hardware.

[11] This is typically used for OSC control

[12] `https://learn.bela.io/using-bela/languages/faust-experimental/`

[13] `https://github.com/grame-cncm/faust/tree/master-dev/architecture/daisy`

2.3. Proposed Solution

A hierarchical namespace can be easily turned into a flat namespace using *fully-qualified names*. This concept exists already in FAUST and is called an *address*, and is used for example in the OSC implementation. Similarly, the same concept can be used for ERBUI compound types. This can be done by introducing new keywords in the ERBUI language to map a control to a fully-qualified name in the FAUST program. The same approach will be used for the ERBB language when binding audio sample data.

The FAUST compiler can transpile its DSP programs into a JSON static representation. This representation, and the use of meta-programming, allows us to generate C++ files with already parts of the computation done, in a way very similar to constant folding in modern compilers. This method is used to bind FAUST program values to Eurorack-blocks controls and similarly for audio sample data.

3. METHOD

3.1. Architecture Files

Making a port to a new language or framework for FAUST relies on "architecture files" as a way to inject code into the generated FAUST C++ code. Architecture files are generic by nature: they are meant to be written once and then dynamically generate at run time whatever is needed by the underlying target.

Typically the FAUST generated function to build the user interface might look like this:

```
virtual void buildUserInterface(UI* ui) {
    ui->openVerticalBox("FLANGER");
    ui->openHorizontalBox("0x00");
    ui->addCheckButton("Bypass", &fCheckbox0);
    ui->addCheckButton("Invert Flange Sum",
    ↪  &fCheckbox1);
    ui->addHorizontalBargraph("Flange LFO",
    ↪  &fHbargraph0, -1.5f, 1.5f);
    ui->closeBox();
    ui->openHorizontalBox("0x00");
    ui->addHorizontalSlider("Speed", &fHslider1,
    ↪  0.5f, 0.0f, 10.0f, 0.00999999978f);
    ui->addHorizontalSlider("Depth", &fHslider5,
    ↪  1.0f, 0.0f, 1.0f, 0.00100000005f);
    ...
}
```

And a GTK+ architecture file would create a Checkbox button on the call to `addCheckButton` and analogously for the rest of the user interface.

Memory allocation is achieved similarly, as the FAUST generated code enumerates what allocations to perform. In particular, the latest version of the FAUST compiler gives additional information on the use of those memory blocks, by instructing how much time they are read or written during a DSP compute step.

3.2. Embedded Systems Constraints

While the previous approach is perfectly valid and simple most of the time for programs running on a desktop computer, it is more problematic on embedded processors.

The embedded system we use has 128K of stack memory, 512K of fast heap memory, and 64MB of slower heap memory, and has a single-core processor running at 480MHz. Therefore the usage of computation resources and memory is very limited compared to a desktop computer.

In the example program above, to bind a `hslider` value to its ERBUI counterpart, we would have to make a small algorithm similar to a simple parser, which is more or less fine but still could use resources that would be hard to reclaim afterward.

This is more problematic for the memory manager, as every time we receive an `allocate` call we need to decide whether to put it in the fast SRAM region or the slower SDRAM region. At the time the `allocate` function is called, and without knowing what will be the future `allocate` calls, we can't make an optimal placement of blocks into memory regions.

For example, consider the following FAUST generated calls:

```
// a delay line accessed only once
memory_manager->allocate (64 * 1024);

 // another delay line accessed only once
memory_manager->allocate (64 * 1024);

// some filter memory accessed multiple times
memory_manager->allocate (12);
```

With a naive heuristic, the two first allocations would fill the entire fast SRAM region, when the last memory block which would benefit from fast SRAM is allocated to the slow SDRAM region.

The latest version of the FAUST compiler generates code that indicates how blocks are accessed, to enable better decision-making, and works in a two-phase manner:

- It first enumerates all memory blocks with their access recurrence and size,
- It then asks the DSP memory manager to allocate those blocks.

This decision problem is very well known in operational research as the *bin packing* optimization problem. Computationally, this problem is NP-hard, but optimal solutions to large problem instances can be produced with sophisticated algorithms, and many approximation algorithms exist as well.

3.3. Moving Computation at Build Time

Solving this problem is however not really adapted to our small embedded processor, and we can observe too that this problem does not need to be solved dynamically either: since FAUST can produce this information when transpiling the DSP file, it is possible to solve this problem at build time rather than run time. This approach makes a huge difference as building the program is done on a powerful desktop computer.

We therefore propose this novel approach in FAUST, to relocate to build-time language bindings, parameter value mappings, and memory management pre-computations, that do not need to execute at run time, to save crucial memory and computation resources on the target embedded platform.

In practice, our system generates C++ code that matches exactly the module that is currently produced, from the ERBUI and ERBB files. We use the `-json` option of the FAUST compiler, which produces a declarative description of the FAUST DSP program, that we can use to generate perfectly-tuned C++ for the embedded platform. An example of the generated code can be found in Appendix A.

At the time when we first implemented the FAUST integration, the new memory manager interface was not available, and the equivalent declarative JSON memory blocks description was not available either. We decided to implement a simple greedy algorithm for now and enhance memory management in future work.

4. IMPLEMENTATION

4.1. Overview

ERBUI ignores most layout information dictated from the FAUST program, such as `hgroup` or `vgroup`, the orientation in `hslider` and `vslider`, or the `style` of a primitive. This allows a clean separation between the ERBUI definitions presentation layer and the FAUST program computation layer.

ERBUI and ERBB introduce a few new keywords specifically for FAUST:

- `faust` represents a lexical scope for definitions specific to FAUST,
- `bind` represents a lexical scope in the FAUST scope to define a binding,
- `init` represents a lexical scope in the FAUST scope to define an overridden initial value for a FAUST primitive,
- `value` represents a scalar value ,
- `address` represents the fully-qualified name of a primitive in a FAUST program,
- `property` represents the property name in an ERBUI compound type.

4.2. Name Mapping

Let's consider the following FAUST program:

```
import("stdfaust.lib");
process = dm.flanger_demo;
```

When running `faust` with the `-json` option we get for example for the Speed pot:

```
{
    "type": "hslider",
    "label": "Speed",
    "address": "/FLANGER/0x00/Speed",
    "meta": [
        { "1": "" },
        { "style": "knob" },
        { "unit": "Hz" }
    ],
    "init": 0.5,
    "min": 0,
    "max": 10,
    "step": 0.01
}
```

The address field `/FLANGER/0x00/Speed` represents the fully-qualified name of the Bypass checkbox in the `dm.flanger_demo`, and is guaranteed to be unique in the whole program.

The following ERBUI source code expresses a binding from a Pot with label SPEED on the Eurorack module, with the Speed FAUST primitive inside `dm.flanger_demo`.

```
control speed Pot {
    faust { bind {
        address "/FLANGER/0x00/Speed"
    }}
    position 6hp, 34mm
    style rogan.6ps
    label "SPEED"
}
```

An example of compound types in ERBUI is a dichromatic LED. It has a `r` and `g` property which denotes the red and green LED part intensity, respectively:

```
control led_bi LedBi {
    faust {
        bind {
            property r
            address "/Phaser/LED Red"
        }
        bind {
            property g
            address "/Phaser/LED Green"
        }
    }
    position 10hp, 80mm
    style led.3mm.red_green
    label "LED"
}
```

4.3. Implicitly-defined Default Binding

If no user-defined bindings are provided for a control, it is defined by the ERBB transpiler, and has the same effect as a user-defined binding with address `"{module}/{control}"` where `{module}` is the name of the module in which the control is defined, and `{control}` is the name of the control. That is, it defines the binding that matches the address FAUST would generate for a primitive in the root function.

For example the FAUST program:

```
// LowPass.dsp

import("stdfaust.lib");

fc = hslider("freq", 1000, 100, 10000, 1);
process = fi.resonlp(fc,1,0.8);
```

Will generate the address `Lowpass/freq` for the `hslider` primitive, and the ERBUI source code:

```
module LowPass {
    control freq Pot {
        position 6hp, 34mm
        style rogan.6ps
        label "FREQ"
    }
}
```

implicitly defining the default binding with the same address `Lowpass/freq`, matching the FAUST generated one.

4.4. Initial Value

When a FAUST primitive in a function of the standard library needs to have an initial value different from the one originally defined in

the library functions, the `init` keyword can be used to override the FAUST-defined value, at module scope:

```
module Flanger {
   ...

   faust { init {
      address "/FLANGER/Delay Controls/Delay
      ↪  Offset"
      value 1
   }}
}
```

4.5. Parameter Value Mapping

The implementation takes into account the minimum and maximum value of a property, as well as its scale (e.g. linear, logarithmic or exponential).

The type of the FAUST primitive is used to provide a scalar value to FAUST following the expectation of the end-user. For example, a `checkbox` type will inform Eurorack-blocks to take the boolean value and convert it to a scalar value, while the `button` type will inform Eurorack-blocks to deliver a scalar value that corresponds to a *rising-edge* of the boolean value.

4.6. Audio Input/Output Mapping

The FAUST DSP defines several buffers per channel, for both input and output. They are mapped to the ERBUI file by order, so that the first audio input channel would map to the first `AudioIn` definition in the ERBUI file, for example.

4.7. Audio Files

FAUST supports only planar (non-interleaved) audio formats, so the ERBB transpiler generates a planar representation suitable for FAUST. This representation can have a performance impact while fetching data, because of its access pattern and the underlying memory chip. But we think this is good enough for a first implementation.

We also ignore the audio file URL described in the `soundfile` metadata and refer to the ERBB definition for that.

4.8. Memory Management

Three regions of memory can be commonly used by the firmware:

- The fast 128KB `DTCMRAM` which runs at full processor speed,
- The 512KB `AXI SRAM` which runs at half of the processor speed,
- The slow 64MB `SDRAM` which is external to the processor.

In general, values that are read and written often, such as filter histories should be written in the fastest RAM. Since this should be small and directly part of the DSP state, we elect the fast `DTCMRAM` to be our stack memory and put the DSP state on it [14].

The rest of the available RAM should be used for everything else.

[14] This is not always the case, unfortunately, as some delay lines might be stored on it.

4.9. Custom Memory Manager

In C and C++, the FAUST compiler produces a class (or a struct in C), to be instantiated to create each DSP instance. The standard generation model produces a flat memory layout, where all fields (scalar and arrays) are simply consecutive in the generated code (following the compilation order).

On audio boards where the memory is separated as several regions (like SRAM and SDRAM) with different access times, it becomes important to refine the DSP memory model so that the DSP structure will not be allocated on a single region of memory, but possibly distributed on all available regions. The idea is then to allocate parts of the DSP that are often accessed in fast memory and the other ones in slow memory.

This is done by using a special `-mem` compilation option which adapts the way the C++ code is generated, and interacts with an external `dsp_memory_manager` object which will distribute the different needed memory zones (like tables or delay lines) on different memory regions.

A custom `dsp_memory_manager` which implements a simple monotonic allocator heuristic has been implemented: since FAUST doesn't allocate or deallocate memory during the computation phase, we can use a greedy algorithm which allocates portions of memory in the `AXI SRAM` as long as they fit in it, and use the `SDRAM` for everything else.

4.10. Code Generation

Both ERBB and ERBUI have non-ambiguous grammars which make them suitable for use with a Parsing Expression Grammar. We generate an Abstract Syntax Tree from the input ERBUI or ERBB file, which goes to an analyzer, and then multiple generators.

The analyzer provides the semantic analysis, and in the context of FAUST, verifies for example that the number of audio inputs and outputs of the FAUST DSP file matches the ones defined in the ERBUI file.

We have two generators for FAUST, one for ERBB and one for ERBUI:

- The ERBB generator generates the C++ FAUST integration layer, as well as audio sample bindings,
- The ERBUI generator generates the C++ UI primitive bindings.

Those generators rely on the fact that the order of definition in the JSON file follows the same order of binding calls (such as `addButton`, `addVerticalSlider` or `addSoundfile`). This allows to avoid to maintain a nested state, as `Box` are `open` and `close` following the program primitive tree structure, which would require some allocations while binding the FAUST primitives to their Eurorack-blocks counterparts.

For example for language binding, we need to adapt between Eurorack-blocks internal storage, which can have various types (like a `float` or boolean) and the FAUST representation which is always a scalar `float` value. For this we have a table of `float*` values, for which each element points to the internal FAUST value representation, and which table is as big as the number of FAUST UI primitives:

```
// In code_template.h of the Erbb Faust
↪  generator
size_t decl_index = 0;
std::array <float*, %faust.widgets.length%>
↪  parameters;
```

`%faust.widgets.length%` is a syntax in the template to indicate the generator to replace this sequence of characters by the numbers of FAUST UI primitives. We deduce this value from the FAUST JSON generated files which allow us to count the total number of widgets in a FAUST program.

We have a representation of the FAUST JSON tree in python, that we deduced from the JSON file:

```
class FaustDsp:
    def __init__ (self):
        self.nbr_inputs = 0
        self.nbr_outputs = 0
        self.widgets = []

class FaustWidget:
    def __init__ (self, type_, address, min_val,
    ↪  max_val, scale):
        self.type_ = type_
        self.address = address
        self.scale = scale
```

Once we have read the FAUST JSON file and created the corresponding python structure, generating the C++ code from the template is just text replacement:

```
# (Simplified generator code for readability)
def generate_module_declaration (self, path,
↪  faust_dsp, module):
    path_template = os.path.join (PATH_THIS,
    ↪  'code_template.h')
    path_output = os.path.join (path, '%s.h' %
    ↪  module.name)

    with open (path_template, 'r',
    ↪  encoding='utf-8') as file:
        template = file.read ()

    template = template.replace
    ↪  ('%faust.widgets.length%', str (len
    ↪  (faust_dsp.widgets)))
    ...

    with open (path_output, 'w',
    ↪  encoding='utf-8') as file:
        file.write (template)
```

Since the order of UI primitives in the JSON file is the same as the FAUST generated C++ code, we can keep track of which index in the `std::array` represents which UI primitive in the FAUST program and which control in Eurorack-blocks. Then, as this information is kept in the generator, the C++ code generation is rather straightforward. For example the FAUST function `addButton` is implemented as:

```
void Adapter::addButton(const char* /* label_0
↪  */, float* zone)
{
    push_parameter (zone);
}
```

And we can see that we completely ignore the label, and only keep the memory zone of where FAUST stores the scalar value. Binding is then straightforward as well:

```
void Adapter::push_parameter(float* zone)
{
    parameters [decl_index] = zone;
    ++decl_index;
}
```

Every time we want to process a new block of audio samples, we need to get the Eurorack-blocks UI control values and set them in the FAUST memory zone before we call the FAUST DSP `compute` function. As we explained above, since we keep a mapping between FAUST UI primitive indexes and Eurorack-blocks control names, we can simply generate the code below. For example, we know that the 0-based index 3 in the FAUST program maps to the control `speed` in Eurorack-blocks.

```
void Adapter::preprocess ()
{
    *module.adapter.parameters[1] = ((0.000000f +
    ↪  1.000000f *
    ↪  (module.ui.invert.position_first () ? 0.f
    ↪  : (module.ui.invert.position_center () ?
    ↪  0.5f : 1.f))));
    *module.adapter.parameters[3] = ((0.000000f +
    ↪  10.000000f * module.ui.speed));
    *module.adapter.parameters[4] = ((0.000000f +
    ↪  1.000000f * module.ui.depth));
    *module.adapter.parameters[5] = ((-0.999000f
    ↪  + 1.998000f * module.ui.feedback));
    *module.adapter.parameters[6] = ((0.000000f +
    ↪  20.000000f * module.ui.flange_delay));
    *module.adapter.parameters[8] = ((-60.000000f
    ↪  + 70.000000f * module.ui.level));
}
```

The above code also contains the parameter value mapping, which takes into consideration the meta minimum and maximum value as defined in the FAUST code. We can also generate the C++ mapping code directly. One will note that the mapping function is affine, and therefore the simplest representation, as we can do pre-calculations in the code generator. This is also useful for output UI primitives like `hbargraph`, as the mapping contains a division, but we can therefore pre-compute it at build time, and keep an affine function as well:

```
// Generated code
void Adapter::postprocess ()
{
    module.ui.led = (0.500000f + 0.333333f *
    ↪  *module.adapter.parameters[2]);
}
```

Sample binding with the FAUST `soundfile` primitives is implemented exactly with the same concepts.

5. CONCLUSION

The resulting FAUST compatibility layer is incredibly thin and matches totally our abstraction without requiring any changes to it. In particular, the clean separation between the presentation layer and computation layer allows immediate benefits for FAUST user every time a new UI feature is developed into ERBUI, with, in most cases, no changes to the ERBB FAUST integration.

We identified and solved two main user stories:

- FAUST users who write new programs benefit from the implicitly-defined default binding, as long as they use primitive labels that can be also ERBUI or ERBB identifiers,
- FAUST users can port past programs without any changes by using our new FAUST-specific keywords.

In our opinion, this represents the "best of both worlds" while providing a cohesive abstraction.

Audio sample usage benefits from the Eurorack-blocks automation that compiles the audio sample directly into the program, simplifying greatly samples management for small samples.

6. FUTURE WORK

The current implementation for memory management expects the FAUST DSP state to fit on the 128K stack of the Daisy platform. This is not the case for DSP modules such as the FAUST reverbs. Additionally, some work has been done on FAUST to include memory management hints into the JSON file, so that our allocator could make better decisions on where to allocate memory, at build time, using a proper bin-packing heuristic. We will address both issues in future work.

The current implementation for `soundfile` doesn't support parts. For this, we will support the Cue/Splice markers in the WAV audio format, which is exactly meant for this purpose. Furthermore, this will allow having continuous data stored in memory rather than just a view on those data, which simplifies the implementation.

7. ACKNOWLEDGEMENTS

The authors would like to thank Eliott Paris who kindly reviewed the earlier version of this manuscript and provided valuable suggestions and comments.

8. REFERENCES

[1] Yann Orlarey, Stéphane Letz, and Dominique Fober, *New Computational Paradigms for Computer Music*, chapter "Faust: an Efficient Functional Approach to DSP Programming", Delatour, Paris, France, 2009.

[2] Romain Michon, Daniel Overholt, Stephane Letz, Yann Orlarey, Dominique Fober, and Catinca Dumitrascu, "A faust architecture for the esp32 microcontroller," in *Proceedings of the Sound and Music Computing Conference (SMC-20)*, Torino, Italy, 2020.

[3] Romain Michon, Yann Orlarey, Stéphane Letz, and Dominique Fober, "Real time audio digital signal processing with Faust and the Teensy," in *Proceedings of the Sound and Music Computing Conference (SMC-19)*, Malaga, Spain, 2019.

9. APPENDIX A

```
// Flanger_erbui.hpp
// Implements name binding, parameter value mapping, DSP computation
// Comments were removed for readability

void Adapter::addButton(const char*, float* zone)
{
    push_parameter(zone);
}

void Adapter::addCheckButton(const char*, float* zone)
{
    push_parameter(zone);
}

void Adapter::addVerticalSlider(const char*, float* zone, float, float, float, float)
{
    push_parameter(zone);
}

void Adapter::addHorizontalSlider(const char*, float* zone, float, float, float, float)
{
    push_parameter(zone);
}

void Adapter::addHorizontalBargraph(const char*, float* zone, float, float)
{
    push_parameter(zone);
}

void Adapter::addVerticalBargraph(const char*, float* zone, float, float)
{
    push_parameter(zone);
}

void Adapter::addNumEntry(const char*, float* zone, float, float, float, float)
{
    push_parameter(zone);
}

// name binding runtime "algorithm"

void Adapter::push_parameter(float * zone)
{
    parameters[decl_index] = zone;
    ++decl_index;
}

// parameter value mapping

void Adapter::preprocess()
{
    *module.adapter.parameters[1] = ((0.000000f + 1.000000f * (module.ui.invert.position_first () ? 0.f : (module.ui.invert.position_center () ? 0.5f : 1.f))));
    *module.adapter.parameters[3] = ((0.000000f + 10.000000f * module.ui.speed));
    *module.adapter.parameters[4] = ((0.000000f + 1.000000f * module.ui.depth));
    *module.adapter.parameters[5] = ((-0.999000f + 1.998000f * module.ui.feedback));
    *module.adapter.parameters[6] = ((0.000000f + 20.000000f * module.ui.flange_delay));
    *module.adapter.parameters[8] = ((-60.000000f + 70.000000f * module.ui.level));
}

void Adapter::postprocess()
```

```
{
   module.ui.led = (0.500000f + 0.333333f *
   ↪  *module.adapter.parameters[2]);
}

// DSP computation

void  Flanger::process()
{
   adapter.preprocess();

   const float* const in[] = {
      &ui.audio_in[0], &ui.audio_in2[0]
   };

   float* const out[] = {
      &ui.audio_out[0], &ui.audio_out2[0]
   };

   auto in_faust = const_cast <float**> (in);
   auto out_faust = const_cast <float**> (out);

   dsp.compute(erb_BUFFER_SIZE, in_faust,
   ↪  out_faust);

   adapter.postprocess();
}
```

ENVELOPE FOLLOWING VIA CASCADED EXPONENTIAL SMOOTHERS FOR LOW-DISTORTION PEAK LIMITING AND MAXIMISATION

Dario Sanfilippo

Independent
Catania, Italy
sanfilippodario@gmail.com

ABSTRACT

In this paper, we discuss the implementation of look-ahead peak limiters using exponential envelope followers and their deployment in multi-band loudness maximisation. Particularly, the limiter's amplitude profiling algorithm uses cascaded one-pole filters to improve derivative continuity and reduce convolution artefacts when multiplying the input signal by the attenuation gain as shown in a total harmonic distortion analysis. The maximiser algorithm allows for two independent maximisation modalities that can be combined: an RMS-based dynamic normalisation of each band with controllable depth to reach equal power, and a global maximisation gain that boosts the individual bands by the same amount. An open-source FAUST implementation of these algorithms is available on Github at the links: `https://github.com/dariosanfilippo/limiterStereo` and `https://github.com/dariosanfilippo/maximiser`.

1. INTRODUCTION

Limiting is a form of dynamic range processing that operates when a signal exceeds a certain amplitude threshold forcing it within boundaries, while it leaves it unaltered when the signal is below the limit. Even though this task can be realised effectively through a clipping function, introducing little or no distortion is key for the processor to perform adequately in a music and audio context. Look-ahead processing is a technique in dynamic range compression and limiting that allows for gain signals to be applied smoothly, hence reducing the distortion introduced by the process. It consists of delaying the input signal by a small amount so that the attenuation curve can reach the target values gradually [1].

Essentially, envelope profiling for the computation of the gain attenuation signal can be performed using two methods: finite impulse response (FIR) filtering, such as rectangular moving average filters, or infinite impulse response (IIR) filtering, such as one-pole low-pass filters (exponentially-weighted averaging units). Both approaches have unique characteristics that can be desirable depending on the specific use case; arguably, the most relevant difference is that FIR systems can reach the target value exactly within a specified period, while IIR systems converge at arbitrary rates. Furthermore, the exponential growth and decay of IIR systems are perceptually more natural compared to the FIR ones.[1] For the reasons above, an FIR system is suitable for brick-wall limiting, that is, limiters that guarantee a strict amplitude boundary for the output signal, whereas IIR systems are more appropriate for the musical context due to their natural-sounding behaviour.

[1] `https://ccrma.stanford.edu/~jos/mdft/Why_Exponentials_Important.html`

In the following sections, we will discuss the key building blocks for the amplitude profiling: the peak-holder and the envelope smoothing units. Then, we will provide an overshooting and total harmonic distortion (THD) analysis for the peak limiter under common settings and broad-spectrum test signals. Finally, we will show how the peak limiter can be deployed in maximisation units through multi-band processing.

2. PEAK LIMITER DESIGN

2.1. Peak Detection Via Cascaded Peak-Holders

The principle of a peak limiter is to detect the amplitude profile of the incoming signal and compute the gain values that scale the input down if above a threshold, or leave it unaltered if below it. For this design, we rely on peak amplitude profiling, using peak-holders. A peak-holder is a system that outputs the input absolute value if larger than or equal to the current out. Otherwise, a timer is set and the system holds the current output for the desired period, after which the current output is updated with the absolute value of the input. The function described above can be used to create a staircase signal tracking the amplitude peaks of the input. The resulting signal can then be smoothed out with exponential smoothers having specific attack and release times to obtain the gain attenuation and control the output amplitude. We will get back to exponential smoothers shortly.

We set the hold period of the peak holder and the look-ahead delay equal to the attack time of the exponential smoother. This way, we allow the smoothers to converge to the target value and synchronise the peaks of the attenuation with those of the input signal. Additionally, besides the attack and release parameters, the limiter has a hold time that extends the hold period of the peak detection section beyond the attack time; this can be particularly useful to improve THD at low frequencies. If we use a single peak-holder, we are not able to detect secondary peaks that appear within the hold period. We can address this problem by cascading N peak-holders, each of them having a hold period that is $1/N$ of the full hold period. This allows for secondary peaks that appear after $1/N$ of the period to be detected. For the design of this limiter, specifically, we use eight cascaded peak-holders, which appear to work well for most cases considering that the non-detected peaks fall within the attenuation curve in the release phase. If the release time is considerably larger than the attack time, as it is often the case, essentially all non-detected peaks will be covered by the release curve and the overshooting will be minimal. See Figures 1, 2, 3, 4, and section 2.4 below. We call the output of the final stage of the peak detection:

$$p_{a,h}[n]. \tag{1}$$

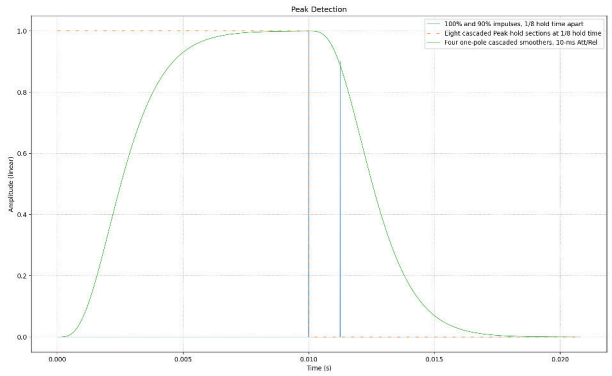


Figure 1: *Edge case for secondary non-detected peaks at 90% of the primary one.*

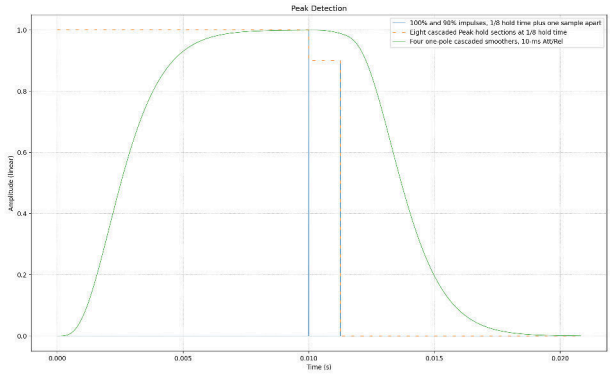


Figure 2: *Effectiveness of cascaded peak-holders to detect secondary peaks.*

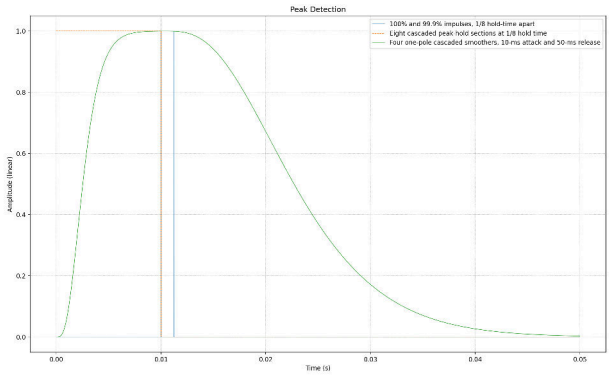


Figure 3: *Edge case for secondary non-detected peaks at 99.9% of the primary one with five times longer release time.*

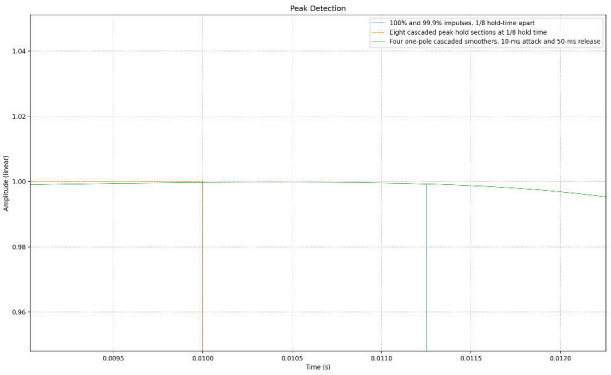


Figure 4: *Edge case for secondary non-detected peaks at 99.9%: close-up.*

2.2. Envelope Following Via Cascaded Exponential Smoothers

Exponential smoothers can be implemented via simple recursive equations representing one-pole systems of the form:

$$s[n] = x[n] + \alpha(s[n-1] - x[n]). \tag{2}$$

The α coefficient determines the pole position of the recursive system, which controls the growth and decay rates.[2] For our design, we use a time constant that is $2\pi\tau$, and we can calculate the coefficient α as:

$$\alpha = e^{\frac{-2\pi T}{\tau}}, \tag{3}$$

where T is the sampling period and τ is the time, in seconds, for the step response to reach $1-e^{-2\pi}$. Single one-pole smoothers for dynamic range compression are discussed extensively in [2]. Envelope following systems with separate attack and release parameters can be implemented as series or parallel configurations of one-pole filters. Giannoulis et al. (2012) investigate branching and decoupled designs for envelope following and improve the THD of the dynamic range processors by smoothing out the transition between attack and release sections: since multiplication in the time-domain corresponds to convolution in the frequency-domain, it is vital that the attenuation curve has continuous derivatives to reduce distortion. However, the attack phase of single one-pole envelope followers shows a sharp onset. We can address this problem by cascading M envelope following sections to guarantee continuity in the derivatives up to order $M-1$, hence producing smooth curves even when transitioning between attack and release phases. The individual envelope following units that we use are branching sections of the following kind:

$$s[n] = \begin{cases} x[n] + \alpha_a(s[n-1] - x[n]), & \text{if } x[n] > s[n-1] \\ x[n] + \alpha_r(s[n-1] - x[n]), & \text{if } x[n] \leq s[n-1] \end{cases}, \tag{4}$$

where α_a and α_r are, respectively, the attack and release sections coefficients. Cascading two or more one-pole filters at the same cut-off results in an overall shifted cut-off at the output stage. [3] suggests a coefficient correction formula to maintain an atten-

[2] `https://ccrma.stanford.edu/~jos/fp/One_Pole.html`

uation of $1/\sqrt{2}$ at cut-off. For M cascaded sections, the cut-off multiplier is given by:

$$C_M = \frac{1}{\sqrt{2^{1/M} - 1}}. \tag{5}$$

Similarly, cascading several one-pole smoothers affects the growth and decay rates, which we can correct using (5) to obtain consistent behaviour when deploying several one-pole systems in series. The calculation of the coefficients for the individual smoothing sections becomes:

$$\alpha_M = e^{\frac{-2\pi T C_M}{\tau}}. \tag{6}$$

The transfer function of the combined sections is:

$$\left(\frac{1 - \alpha_M}{1 - \alpha_M z^{-1}}\right)^M. \tag{7}$$

Since the coefficient is the same for each section, cascading M one-pole smoothers only requires $2M$ extra sums and M extra multiplies. We call the output of the final stage of M cascaded one-pole smoothers:

$$s_{a,h,r,M}[n]. \tag{8}$$

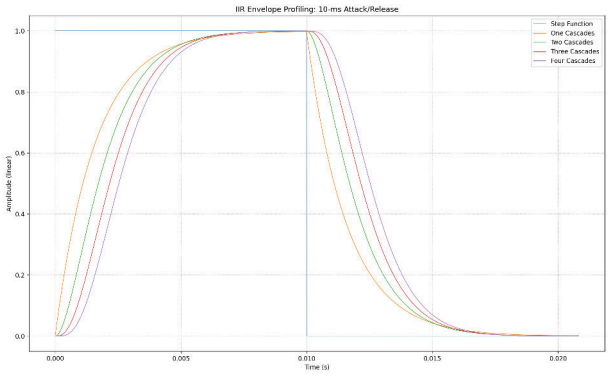


Figure 5: *Step response of cascaded exponential smoothers.*

2.3. Attenuation Gain Computation

As discussed in [4], since the input signal must remain unaltered when its peaks are below the threshold, the computation of the attenuation signal requires a min or max function to clip the gain at a neutral level when needed. These clipping functions produce a discontinuity in the derivative of the signal while transitioning from the neutral to the attenuating values and vice versa. If l is the limiter ceiling, one option to compute the attenuation gain is:

$$g_{a,h,r,l,M}[n] = min\left(1, \frac{l}{s_{a,h,r,M}[n]}\right), \tag{9}$$

which must be smoothed out subsequently to remove any sharp edges in the signal. Alternatively, we can clip at the stage of peak detection, before applying the smoothing processing, leading to:

$$q_{a,h,l}[n] = max\left(l, p_{a,h}[n]\right), \tag{10}$$

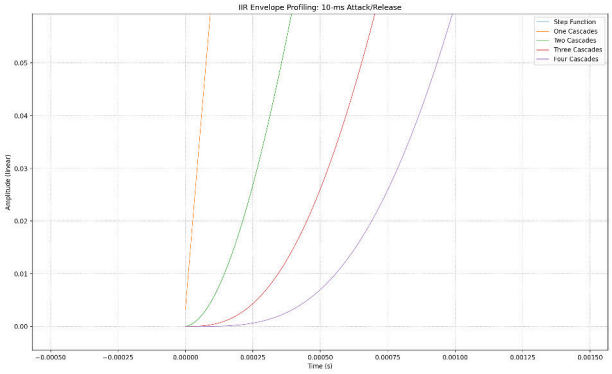


Figure 6: *Step response of cascaded exponential smoothers: close-up of the attack phase.*

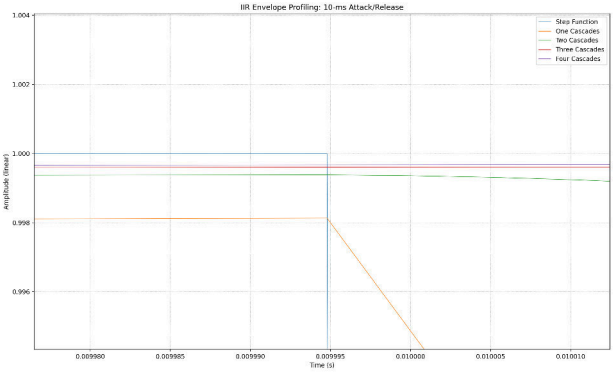


Figure 7: *Step response of cascaded exponential smoothers: close-up of the attack-to-release transition.*

and use the resulting signal as input for the first stage of the cascaded smoothing. Besides no longer requiring a smoothing filter for the final gain, a key difference with this approach is that the envelope smoothers, when transitioning between neutral and attenuating gain values, charge from – and discharge to – the ceiling threshold rather than the current peak, hence using the attack and release times more effectively. See Figure 8 and 9 for a comparison. Also note that, assuming $l > 0$, there is no need to check for division by 0. Then, the attenuation signal is:

$$g_{a,h,r,l,M}[n] = \frac{l}{s_{a,h,r,M}[n]}. \tag{11}$$

2.4. Overshooting and Total Harmonic Distortion Analysis

The attack phase step response of M τ-constant cascaded exponential filters, that is, filters that charge by $1 - e^{-1}$ at a specified time $t \geq 0$, can be determined using the following closed-form equation:

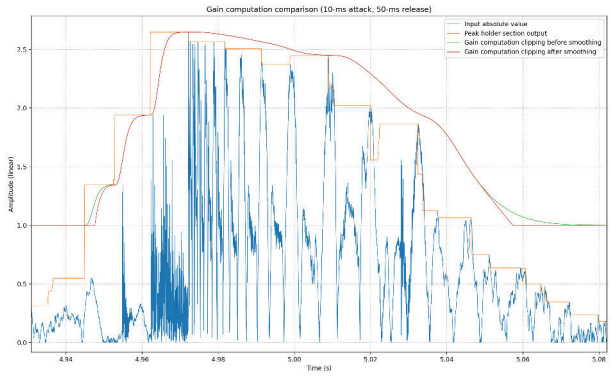


Figure 8: *Comparison of envelope followers for gain computation: transitions between neutral and attenuating values with a 0 dB ceiling. Attack and release times are .01 and .05 second; 0 hold time. Note that the input is delayed by the attack time to synchronise the gain outputs.*

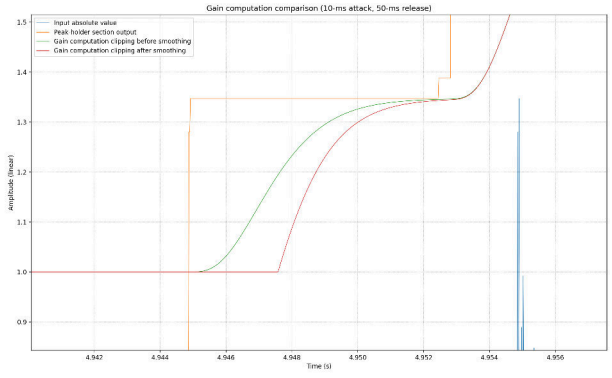


Figure 9: *Gain computation comparison: close-up of the attack phase.*

$$A_{\tau_a,M}(t) = 1 - e^{\frac{-t}{\tau_a}} \sum_{m=0}^{M-1} \frac{(\frac{t}{\tau_a})^m}{m!}, \tag{12}$$

where τ_a is the response time in seconds.[3] We can extend the equation to describe a $2\pi\tau$-constant system and include the coefficient correction from (5). We have that:

$$A_{2\pi\tau_a,C_M}(t) = 1 - e^{\frac{-2\pi t C_M}{\tau_a}} \sum_{m=0}^{M-1} \frac{(\frac{2\pi t C_M}{\tau_a})^m}{m!}. \tag{13}$$

Coefficient-corrected series IIR smoothers build up more slowly but converge more quickly. The envelope follower for the proposed limiter uses four cascaded branching sections such as the ones described in (4). From (13), we can see that the peak of the attack step response of the amplitude profiler is:

$$K_4 = A_{2\pi\tau_a,C_4}(\tau_a) \approx 0.99966855948853062. \tag{14}$$

Hence, for the attack phase, we can predict a maximum overshooting in dB of:

$$|20\log(K_4)| \approx 0.002879332894, \tag{15}$$

which is independent of the attack time. The release phase step response at time $t > \tau_a$ is:

$$R_{2\pi\tau_r,C_M}(t) = K_M e^{\frac{-2\pi(t-\tau_a)C_M}{\tau_r}} \sum_{m=0}^{M-1} \frac{(\frac{2\pi(t-\tau_a)C_M}{\tau_r})^m}{m!}, \tag{16}$$

where τ_r is the release response time. Considering the worst-case for the non-detected secondary peaks discussed in 2.1, we can calculate the maximum overshooting in the release phase using:

$$R_{2\pi\tau_r,C_M}\left(\tau_a + \frac{\tau_a}{N}\right), \tag{17}$$

where N is the number of cascaded peak-holder sections at $1/N$ hold time. In limiters, it is common for the release time to be considerably larger than the attack time. If we choose τ_r as a factor of τ_a, we make the overshooting dependent only on the factor itself. For example, by setting τ_r to twice, five times, or ten times τ_a, we guarantee the overshooting to be below the following dB values regardless of the attack time:

$$\left|20\log\left(R_{2\pi\tau_{2a},C_4}\left(\tau_a + \frac{\tau_a}{8}\right)\right)\right| \approx 0.1217966284, \tag{18}$$

$$\left|20\log\left(R_{2\pi\tau_{5a},C_4}\left(\tau_a + \frac{\tau_a}{8}\right)\right)\right| \approx 0.00750023597, \tag{19}$$

$$\left|20\log\left(R_{2\pi\tau_{10a},C_4}\left(\tau_a + \frac{\tau_a}{8}\right)\right)\right| \approx 0.003213466763. \tag{20}$$

Studies show loudness just noticeable difference to be approximately 0.1 dB or higher [5]. Based on that information, we can scale down the output of the limiter by the maximum overshooting values to guarantee a clip-free output and a negligible loudness loss for settings where the release time is twice the attack time or higher. Alternatively, we can deploy a 0 dB hard-clipping function at the output of the limiter and maximum ceiling to avoid loudness losses while increasing the overall THD by minimal amounts, i.e.: 0.5 % or less for release times that are twice the attack time or higher. Although cascaded exponential smoothers are not suitable for brick-wall limiting and clip-free outputs at maximal loudness levels, the THD measurements in Figure 10 show a significant reduction in the overall degradation caused by ring modulation distortion while the loudness loss is essentially imperceptible. The final output of the limiter is:

$$y_{a,h,r,l,M}[n] = g_{a,h,r,l,M}[n]x[n - a/T], \tag{21}$$

which shows that the limiter introduces a delay in the signal that is equal to the attack time. Multi-channel limiters can be realised by feeding the peak detection unit with the maximum peak of the individual channels to compute a global attenuation signal that applies to all channels.

[3] `https://signalsmith-audio.co.uk/writing/2022/limiter`

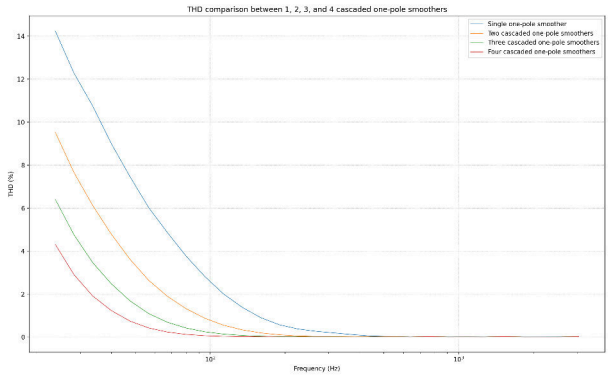


Figure 10: *Total harmonic distortion comparison between single, two, three, and four cascaded exponential smoothers. The input test signal is a sinusoid at 0 dB. The limiter's pre-gain is 60 dB, the attack time is .001 second, the hold time is 0, the release time is .1 second, and the output ceiling is 0 dB. THD values are calculated every quarter octave starting from 20 Hz. Specifically, the measurement is carried out as the ratio between the RMS of the delay-compensated input and output signals difference, and the RMS of the input signal.*

3. LOUDNESS MAXIMISER DESIGN

3.1. Multi-Band Processing

Look-ahead limiters alone can provide significant loudness increase when operating with boosted signals, although degradation and background noise are likely to emerge when the amplification is particularly high. One way to address this problem is multi-band processing, that is, the amplification and limiting of individual spectral bands so that artefacts can be reduced by distributing the stress of the boosting evenly: some spectral bands in a signal are typically weaker than others and can be amplified more before degradation becomes significant.

This maximiser's implementation uses an eight-way Linkwitz-Riley fourth-order crossover [6], which is available in FAUST's Standard Library. In this early implementation of the maximiser, the bands spacing is fixed with a single-octave width starting from 40 Hz. Furthermore, this version has global parameters for all of the bands although future improvements may deploy individual band settings for more versatile configurations. Each band is connected to a limiter, and their outputs are summed into a final limiter stage that controls the output ceiling. Particularly, the limiters connected to each band have a fixed ceiling of 0 dB, while the final output ceiling can be adjusted to arbitrary values. Thus, the overall delay of the maximiser is twice the combined attack and hold times. Also note that, while the release parameter can be set independently for each band in future versions of the maximiser, the attack and hold times must be the same for all bands to preserve phase alignment and a flat magnitude response when summing out the bands.

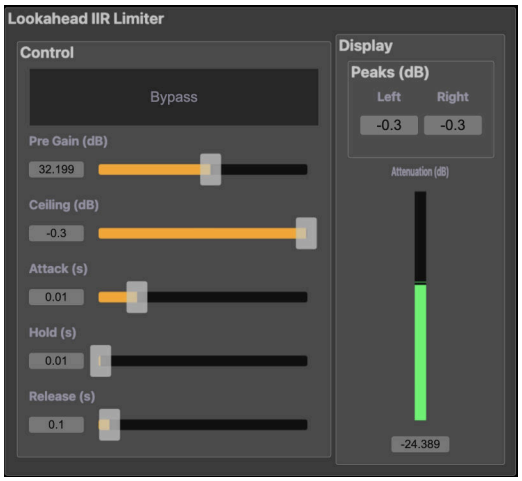


Figure 11: *Limiter stereo UI.*

3.2. Maximisation Modalities

Essentially, the processor has two modalities for multi-band maximisation that can operate in conjunction. On one hand, the maximiser offers the possibility to dynamically normalise the bands so that they have the same level approximately; on a sample-by-sample basis, the RMS of the loudest band is the reference value used to calculate a gain factor to bring all bands at the same level. The RMS calculation is carried out with $2\pi\tau$-constant one-pole filters and a one-second response time. Connected to this feature is a normalisation depth parameter that sets the maximum gain amount for normalisation. Assuming that the input signal without amplification is below 0 dB, the dynamical normalisation process does not set the limiters of the individual bands into an operational mode; hence, this process is free from degradation except for the limiting provided by the final limiter stage when the sum of the individual bands exceeds the ceiling. Alternatively, the maximisation can be performed by applying a gain amplification to the input signal, hence boosting all of the bands equally. The main difference is that dynamical normalisation provides maximum individual amplification with minimum degradation. On the other hand, the global gain amplification results in higher degradation for the predominant bands while keeping the overall spectral weights closer to the original signal.

Figure 12: *Maximiser mono UI.*

4. CONCLUSION

In this paper, we have discussed the implementation of look-ahead peak limiters deploying cascaded exponential smoothers for envelope following. Besides the natural-sounding behaviour of exponential smoothers, measurements show that implementing cascaded sections of one-pole envelope followers improves the THD when used in look-ahead limiting. Furthermore, IIR filters offer an alternative to moving average filters and reduce memory requirements, while calculations show that the overshooting of cascaded IIR filters is negligible for most musical applications. Finally, the paper shows an implementation of the peak limiter design in multiband maximisers offering two non-exclusive modalities: dynamical band normalisation, for low-degradation maximisation, and homogeneous signal boosting to preserve, to some extent, the original bands weights. Future work on the limiter will include the implementation of adaptive mechanisms for automatic adjustments of the parameters and optimal THD and recovery time. We will include the possibility to switch between amplitude profiling in the linear or log (dB) domains. Furthermore, the author will explore a creative applications of the limiter in his work on music feedback systems, specifically using the limiter with non-standard parameter settings as a stability processor in self-oscillating systems.

5. ACKNOWLEDGMENTS

I would like to thank Geraint "Signalsmith" Luff for providing inspiration and insights during the development of this work.

6. REFERENCES

[1] Udo Zölzer, Xavier Amatriain, Daniel Arfib, Jordi Bonada, Giovanni De Poli, Pierre Dutilleux, Gianpaolo Evangelista, Florian Keiler, Alex Loscos, Davide Rocchesso, et al., *DAFX-Digital audio effects*, John Wiley & Sons, 2002.

[2] Dimitrios Giannoulis, Michael Massberg, and Joshua D Reiss, "Digital dynamic range compressor design—a tutorial and analysis," *Journal of the Audio Engineering Society*, vol. 60, no. 6, pp. 399–408, 2012.

[3] Vadim Zavalishin, "The art of va filter design," *Native Instruments, Berlin, Germany*, 2012.

[4] Perttu Hämäläinen, "Smoothing of the control signal without clipped output in digital peak limiters," in *in International Conference on Digital Audio Effects (DAFx)*, Hamburg, Germany, 2002, NordiCHI, pp. 195–198, DAFx.

[5] Scott G. Norcross, Gilbert A. Soulodre, and Michel C. Lavoie, "The subjective loudness of typical program material," *Journal of the Audio Engineering Society*, October 2003.

[6] Siegfried H Linkwitz, "Active crossover networks for noncoincident drivers," *Journal of the Audio Engineering Society*, vol. 24, no. 1, pp. 2–8, 1976.

STATIC PERFORMANCE ANALYSIS OF FAUST PROGRAMS USING DATAFLOW MODELLING

Jaime Koh *

School of Computing
National University of Singapore
Singapore
jaimekoh@u.nus.edu

Bruno Bodin

Yale-NUS College
National University of Singapore
Singapore

ABSTRACT

The processing capabilities of modern System-on-Chips allow artists to push the boundaries of digital art by enabling the implementation of complex real-time signal processing applications. The Faust framework has simplified the process of implementing signal processing applications for these systems. With the Faust language, it is possible to generate implementations for a wide variety of target contexts, allowing users to take a "write once, implement everywhere" approach. Nonetheless, this variety of hardware also comes with the issue of determining what kind of processing capability is required for a given project.

Evaluation tools of Faust currently rely on runtime analysis to measure the performance of implementations on target hardware; this method can be time consuming and requires the user to already own the hardware they wish to evaluate.

Instead, we attempt to address this problem by considering static performance analysis. We propose to use a dataflow model of computation to represent Faust programs, and to predict performance for specific target hardware. This approach could provide useful heuristics in the future for users to decide which hardware is needed to reach their required performance.

1. INTRODUCTION

Computers and Embedded systems are a significant enabler of audiovisual digital art [1]. In the early 90s, artists started to use increasingly affordable System-on-Chip (SoC) [2]. Nowadays, a variety of systems are available across a range of performance capabilities and price points; they often contain highly efficient devices such as Graphic Processing Units (GPUs), Digital Signal Processors, and Field Programmable Gate Arrays (FPGAs) that pave the way for new generations of digital art movements [3]. It is therefore becoming increasingly viable for artists who wish to take advantage of specialised processor capabilities to purchase these systems.

Nonetheless, programming these modern systems can be tedious, often requiring some level of engineering expertise, resulting in high barriers to entry.

The Faust programming language addresses this problem by providing a Domain Specific Language (DSL) for audio signal processing, and a compiler framework that can be used to generate programs in a wide variety of languages compatible with various hardware architectures. Users can therefore take a "write once, implement everywhere" approach for the targets that the Faust compiler supports.

* Corresponding author

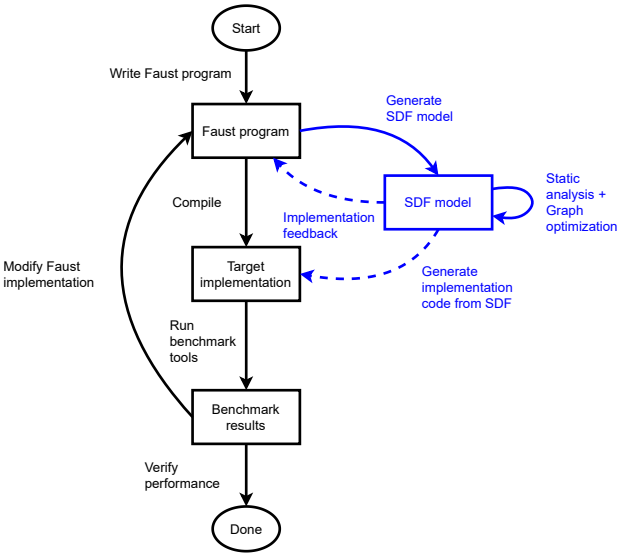


Figure 1: Overview of workflow with/without SDF modelling

While programming these systems has been simplified, the variety of devices available and across a range of costs led us toward the following questions:

- What hardware is needed for a particular (Faust) program?
- Could the (Faust) programming framework point the user towards the minimal required target?

Performance metrics exist to indicate whether a system performs well or not. In the case of audio processing, to match the sampling rate of the system, it is necessary to process samples at a rate of at least 44.1kHz — this is known as the throughput of the system. Additionally, the time taken for the system to produce output samples from input samples also matters — this is known as the system's latency. In practice, the time taken to process a sample needs to remain within 10% of the fraction of time that the sample represents. Furthermore, the latency of an audio processing system is largely influenced by the buffer size used. Factors surrounding a system's implementation of audio processing, such as its audio card, analogue-to-digital/digital-to-analog converters, and communications protocol, can also impact its overall latency. In the context of digital musical instruments, the acceptable upper bound for latency is expected to lie between 10ms to 30ms [4, 5, 6]; exceeding that bound would make it increasingly likely for a user to detect this latency and degrade their performance with the instrument.

Faust includes benchmarking tools to measure the throughput of a given Faust program, however, its primary purpose is to evaluate the performance of different compilation schemes used by the Faust compiler [7, 8]. While this provides insight into the overall performance of the given Faust program, the measured results are dependant on the hardware that the benchmark tools are run on. Evaluating the performance of the same program on different hardware and for different configurations would therefore require additional testing with implementations on the corresponding target hardware. There are shortcomings to this approach. Firstly, re-implementing and running benchmark tests on multiple target hardware can be time consuming. Moreover, this form of testing requires the user to already own the hardware they intend to test their program on; resulting in unnecessary effort and expenditure should the program prove to be unsuitable for the given hardware.

Instead, we propose to rely on static analysis to estimate the performance of a program. Our objective is to divorce performance evaluation from actually running the Faust program on the target hardware.

In the case of general purpose languages (e.g. C or Java) and modern processors (out-of-order CPU with memory caches), due to their high complexity, instrumentation and simulation are preferred to static analysis. However, previous works have shown that when either of these conditions are relaxed by using more deterministic hardware, or simply reducing the expressiveness of the programming language, successful performance estimation can be reached [9].

Fortunately, as a domain specific language, the Faust language has a reduced expressivity and thus has the potential to be successfully used in the context of static performance analysis. For example, there are similarities between the Faust programming language and dataflow models of computation such as Synchronous dataflow graphs [10] (SDFGs).

These models are commonly used in the context of digital signal processing (DSP), describing applications as a set of actors communicating via First-in-First-out (FIFO) buffers. Execution times and communication rates between actors are fixed — this strongly facilitates static analysis, including performance prediction. Numerous frameworks have been proposed that effectively predict the correctness and performance of SDF programs [11, 12].

The decision not to use a dataflow model in Faust's design has been discussed by Orlarey et al. [13] — the complexity and overheads involved in implementing a dataflow model were highlighted as its key shortcomings. Nonetheless, there are benefits to using a dataflow representation as a model for program analysis rather than a runtime model.

Once we model a Faust program as an SDFG, by reusing these existing frameworks, we could anticipate the performance of Faust programs on various hardware without having to actually implement the program on the target hardware, thereby avoiding the shortcomings of the current benchmarking methodology. This approach also allows the performance of Faust programs to be statically evaluated as SDFGs, providing the user with information that may help with optimizing their programs.

In service of this goal, we made the following contributions:

- Describe a model transformation from Faust to SDFG,
- Implemented a compilation scheme that performs this transformation,
- Demonstrate its advantages using several use cases.

We start in Section 2 by providing an overview of Faust and the methods currently available for benchmarking programs. We then present the nomenclature of the dataflow model of computation and, accordingly, how Faust programs can be expressed as an SDF graph in Section 3. As mentioned before, there are benefits to be gained from utilising an SDF model of Faust programs in the process of generating target implementations. In Section 4, we explain the various performance analysis results that are obtainable from SDFGs and show how this model could aid the developer in determining which implementation is most suitable for their given program. We also speculate on potential future applications of the SDF model in the context of implementing Faust programs. Naturally, our proposed use of SDF modelling is not without its shortcomings; the limitations of this approach as well as the potential challenges that might be faced in developing this approach are highlighted in Section 5.

2. THE FAUST LANGUAGE

Faust is a programming language meant to facilitate the implementation of real-time audio signal processing and audio synthesis programs. It is a functional programming language consisting of a signal processing semantic for describing block diagram algebra [13]. It is one of several languages for computer music; these are programming languages that tend to prioritize aspects such as timing, concurrency, and collaboration to facilitate audio synthesis and musical composition [14]. In contrast to most other computer music languages, however, the Faust programming language centres around mathematically expressing signal flow and operations.

2.1. Syntax

As we will be presenting some basic examples of Faust programs in this paper, we provide a brief overview of the language in order to familiarize the reader with the necessary subset of its syntax and semantics. We encourage readers to look to Orlarey *et al.*'s paper for a formal specification of the syntax and semantics of the Faust language [13].

2.1.1. Signals

A signal consists of a stream of sample values that are produced at a fixed sampling rate (e.g. 48000Hz). As the Faust language is primarily concerned with describing digital signal processors, it typically takes in signals as inputs ("input signals"), and produces signals as outputs ("output signals"). The input signals tend to be audio signals, such as those produced by a microphone or an audio file.

2.1.2. Primitives

The Faust language includes several primitives that operate on signals. For the purpose of this paper, we list only three types of primitives:

- Arithmetic primitives ($+, -, *, /, \%$)
- Fixed delay ($@$)
- Identity primitive (_)

Arithmetic primitives consist of five operators — $+, -, *, /, \%$ — which correspond to the addition, subtraction, product, division, and modulo operators respectively. Fixed delay, denoted by '@',

is used to denote an n sample delay between the arrival of an input signal and the production of its output signal. Finally, the identity primitive, denoted by '_', simply refers to the identity function for its input signal.

2.1.3. Composition Operations

To describe the flow of signals between Faust primitives, Faust uses five binary composition operators. We list two of these for the purposes of the examples used in this paper.

- Sequential (`:`)
- Parallel (`,`)

By composing signals with primitives, the Faust language can be used to describe signal processing programs. We demonstrate this with an example Faust program:

```
process = _,2 : / : _,1 : +;
```

Listing 1: A simple Faust program

The program described in Listing 1 can be broken up into a four-step sequence (`s1:s2:s3:s4`). The first step is a parallel composition of the input signal (as identity _) and the numerical value `2`. The second step applies the division operator to the parallel signals, thus dividing the input signal by 2. The third step is again a parallel composition consisting of the output of the division operator (as identity _) and the numerical value `1`. Lastly, the addition operator is applied to the parallel signals. Mathematically, if we denote i the input sample, then this simple program computes $(i/2) + 1$ for each i to compute the corresponding output signal.

Arithmetic primitives and composition operators, along with those not introduced in this section, are thus used to describe signal processing programs. The Faust compiler then compiles the given Faust program to the specified target implementation. Faust provides a benchmark tool to evaluate the performance of the compiled program — the details of which will be covered in Section 2.2.

2.2. Benchmark Tools

Faust provides an ALSA GTK application (CoreAudio for OSX) that can be used to measure the throughput of the given Faust program in terms of megabytes of samples that the program can process per second (MB/s) [8]. This is done by taking a fixed number of timing measurements, converted from processor cycles, which are subsequently used to compute the MB/s achieved by the program. This throughput performance indicator can be used to estimate an upper bound on the speed of a program [7].

Faust includes a suite of benchmark test programs in its distribution — this was initially used to measure the performance impacts of two compilation schemes that added various forms of parallelisations to Faust programs [7]. Subsequently, additional benchmark tools were implemented to evaluate the performance of WebAssembly code generated from Faust programs [15].

While the benchmarking tools can be used to estimate the performance of a given Faust program [7], the results gained are dependent on the characteristics of the machine it is run on. Related to this, the average of several measurements have to be taken as well in order to ensure that the results are stable.[1] The entire benchmarking process can therefore be a time-consuming and subjective process.

[1] As mentioned in the readme of the Faust benchmark tool: `https://github.com/grame-cncm/faust/tree/master-dev/benchmark`.

3. PROPOSED METHODOLOGY

In this section, we first introduce our methodology, then we recall the SDF model that will be used to represent Faust programs. Following which, we explain how signal graphs generated by the Faust compiler are transformed into equivalent SDF representations.

3.1. Overview

3.1.1. Existing Workflow

Figure 1 illustrates an overview of the workflow of writing, generating, and evaluating the performance of a program with Faust. The black portions of the chart indicate the workflow given the current set of benchmarking tools. After writing a program in Faust, users would then compile the program by specifying a target architecture to obtain a program suitable for this target. To evaluate the performance of their code, they would have to run their program using the benchmark tool to obtain the latency and/or throughput of the program executed on the given hardware.

If the compiled implementation does not meet their constraints, they would then have to modify the Faust program before restarting the process — this is indicated in Figure 1 by the black edge from "Benchmark results" to "Faust program". Ultimately, if these modifications were not deemed to be sufficient, the user would need to target a different hardware, and repeat the process.

3.1.2. Our Approach

The blue components in Figure 1 indicate the modified workflow from utilising an SDF model of the Faust program. Prior to compiling the Faust program for a given implementation, it is first modelled as an SDF graph, as indicated by the edge labelled "Generate SDF model" in Figure 1. We would then be able to take advantage of static analysis techniques for SDF graphs to predict its performance prior to running the program. Shortcomings in performance could be identified and corrected at this stage; this is indicated by the feedback loop on the "SDF model" component. On top of avoiding the repeated steps of implementation and running the benchmarking tools, these performance evaluation could be performed for several kind of hardware, helping to identify which device would be the best fit for any particular application.

For example, instead of measuring the latency or the throughput of a particular Faust program at run-time, we could utilise latency and throughput computation techniques designed for SDF graphs. These would allow us to predict the performance of a Faust program without having to implement it on the target machine. Furthermore, we would not experience the same variability as in benchmarking results. In the following sections, we introduce the SDF model followed by our transformation methodology.

3.2. Data-Driven Models

The dataflow computational model represents programs as directed graphs. The nodes of the graph are actors that represent operations to be executed on input data to produce output data. The edges connecting these nodes are channels, which represent first in, first out (FIFO) buffers through which tokens of data ("tokens",

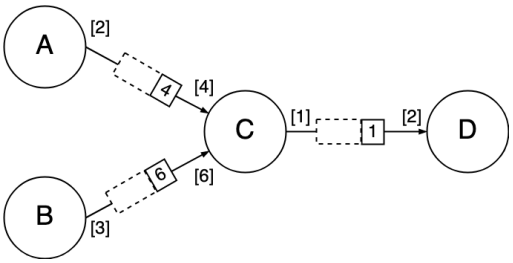


Figure 2: Example of a Synchronous Dataflow Graph.

in short) flow between actors [16]. These channels can be specified with an initial number of tokens ("initial tokens") in their FIFO buffers before the beginning of actor executions. In the context of signal processing programs, a single token would refer to a single sample of a signal. In this model, actors can only be executed when the required number of tokens are available on their input channels, thereby allowing multiple actors to execute in parallel without relying on a centralized control mechanism. In this manner, the dataflow model is a data-driven model, where valid executions of an actor are determined by the availability of the prerequisite data on its input edges.

Synchronous Dataflow (SDF) [10] is a subset of the dataflow computational model. The distinguishing feature of SDF is that the data production and consumption rates of each actor are constant and statically defined [10]. Figure 2 illustrates an example of a simple SDF graph made up of 4 actors. The production and consumption rates are denoted by the numbers in square brackets above the edges going out and into the actor. Actor C, for example, consumes 4 tokens produced by A and 6 tokens produced by B in order to produce 1 token onto the channel connecting C to D. The boxes lying on the edges denote the FIFO buffers through which the tokens flow between actors, while the numbers in the solid edged rectangles refer to the number of initial tokens in the corresponding buffers. Actor C, for example, can execute prior to the first executions of A and B as it already has the sufficient number of initial tokens on its incoming channels. Actor D, on the other hand, will not be able to execute until C executes at least once as it only has 1 initial token in its input channel when it requires 2 tokens to execute. We can thus see how the edges between actors model data dependencies.

An advantage of the SDF model is that, with statically defined production and consumption rates, it is possible to compute a valid execution schedule for the modelled application. This allows us to statically determine several properties of programs modelled using the SDF model:

- Will the program deadlock?
- Is it able to infinitely execute its actors while storing a bounded amount of data?
- How frequently can actors execute?

These analyses would not be possible with general dataflow models [17]. SDF is therefore commonly used to describe DSP applications (e.g. LTE encoders [18], Deep Neural Networks [19]) or to express parallelism for implementation on specialized hardware (e.g. GPU [20], FPGA [21]). By modelling Faust programs using the SDF model, we can apply the same static analysis techniques on signal processors described with the Faust language.

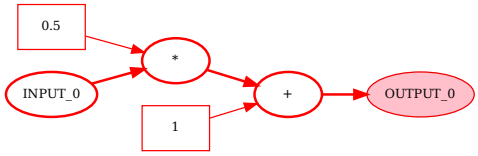


Figure 3: Signal expression generated from Listing 1

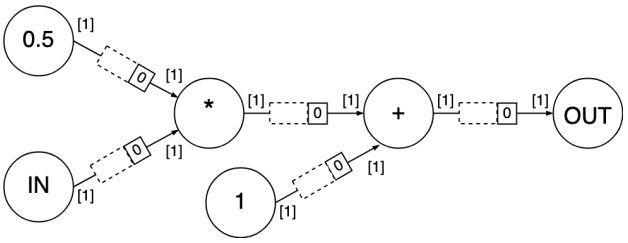


Figure 4: SDFG representation of the signal expression from Figure 3

3.3. Model Transformation

As mentioned in Section 2, Faust programs are written in a high-level functional style which can then be compiled to a variety of target languages and platforms. Part of the process of compiling Faust programs involves converting high-level block-diagram descriptions into equivalent signal expressions, which consist of nodes that consume and produce input and output signals along edges that denote the flow of signals between nodes.

The signal expression in Figure 3, for example, shows how an input signal is multiplied by a constant signal of value 0.5 and subsequently incremented by a constant signal of value 1 in order to produce the output signal. These signal expressions form the basis of generating the SDF model of the corresponding Faust program. We start by describing the ideal scenario in which a signal expression can be directly translated into an equivalent SDF representation before listing out the various modifications that need to be made for signal expression components that do not directly fit into the SDF model.

3.3.1. Generating Actors and Channels

The process of generating an SDF representation of a Faust program starts with generating a signal expression using the Faust compiler. It is worth noting that signal expressions closely resemble SDF graphs. In particular, they resemble a special type of SDF where the production and consumption rates of all actors are equal to 1 — these are known as Homogeneous Synchronous Dataflow graphs (HSDFG).

Looking at the signal expression in Figure 3, each node either provides a constant signal value for the operator it is connected to (such as '0.5' or '1'), or represents an operation to be executed on its input data (such as '$*$' and '$+$'). Given that operations act on a single sample per execution, they are modelled with a fixed consumption and production rate of 1 unit of data per execution. Each operator and each constant are thus modelled as HSDF actors with channels interconnecting them in the same manner that they are connected in the signal expression.

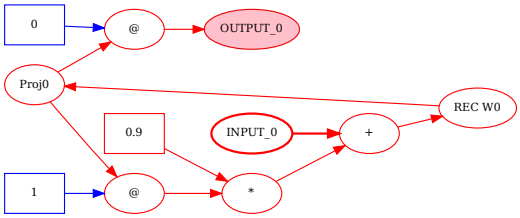


Figure 5: Signal expression of *one-pole filter*

An exception to this are the 'INPUT' and 'OUTPUT' nodes, which represent the source and destination of the input and resulting output signal, respectively. For simplicity, we keep the production and consumption rates of these nodes at 1. However, these values can be changed to reflect different sampling rates, for example. We also model these nodes as actors.

By following these steps, the signal expression in Figure 3 can be modelled as the SDFG shown in Figure 4.

Unfortunately, signal expressions do not always map onto the SDF model as neatly as they do in the above example. In the remainder of this section, we describe the additional modifications involved in adapting signal expressions to the SDF model.

3.3.2. *Modelling Delay in SDF*

Operators in the signal expressions generated by Faust are not necessarily directly compatible with the SDF model; the delay operator is one such operator that we had to adapt to fit the SDF model. Figure 5 illustrates the signal expression of a one-pole filter implemented in Faust. As we can see, there is a cycle of nodes passing through `*`, `+`, `REC_W0`, `Proj0`, and `@`. An equivalent SDF would deadlock, as each node would require the previous one to be executed. In such a case, initial values must be found in one of the cycle's buffers to avoid any deadlock.

The `@` node denotes a fixed delay operator. As mentioned in Section 2, the fixed delay operator is one of the Faust primitives. In a signal expression, it typically consists of an input signal and a node representing a constant integer signal denoting the delay length — a simplified version of the delay operator is illustrated in Figure 6. The delay operators in Figure 5 therefore delay sending the result of the addition operator to their respective output channels by 0 and 1 sample respectively. For this reason, in the first iteration, the product operator ($\star$) will take 0 as an input sample.

Within SDF, initial values are also commonly known as "delay", and perform similarly to delay operators of Faust. We thus replaced delay operators with channels initialized with n initial tokens of data of value 0, as shown in Figure 7. This is performed for each delay operator in order to model its behaviour in the resulting SDF graph. One caveat of this solution is when the delays within Faust programs are interactively controlled by the user with UI components, this case will be discussed in Section 5.

3.3.3. *Timing Constraints*

To perform any kind of timing analysis on SDFG, we need to determine the execution time for each actor as well as their resource constraints limiting their execution or communication patterns. These timings are related to the targeted architectures, and will change depending on the device.

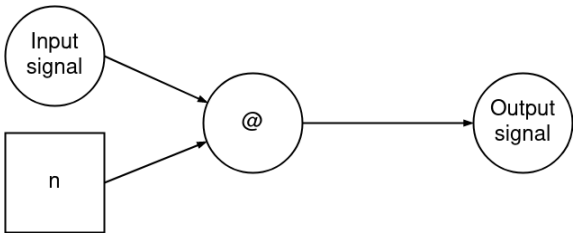


Figure 6: Signal expression with delay of n samples

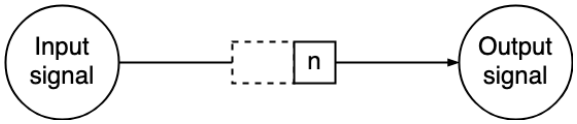


Figure 7: SDF-equivalent modelling delay function

For example, in the context of FPGAs hardware, tools such as the FloPoCo framework have shown that it is possible to automate arithmetic operator implementations with fixed timing guarantees at various frequencies [22].

In different hardware, where fixed guarantees are not achievable, it is sufficient to set the execution time of each actor according to their respective operator's worst-case execution time (WCET). The execution times of the actors are thus set to any duration required by the operator to return a result after receiving its required input data.

However, due to memory accesses and the effect of caches, in devices such as CPUs, the communication time between actors can be much more important than the execution time of individual actors. In this context, rigorous hardware architecture models must be considered during the performance analysis [23].

4. PERFORMANCE ANALYSIS

Modelling Faust programs as equivalent SDF representations allows us to apply the static analysis techniques available to SDFGs to programs implemented in Faust. In this section, we present two performance analysis use cases from modelling Faust programs as SDFGs: throughput, and delay. We provide examples of how these performance metrics can be used in the context of a Faust program that implements an echo effect on an input signal. The signal expression of this program, eponymously named *Echo*, is shown in Figure 8. Using the methodology explained in Section 3.3, we generated an SDF from the signal expression of *Echo*. In the process of doing so, we found that one of the delay primitives in Figure 8 had an argument that was determined by the value of a "hslider" component — the value of the signals from this component are controlled by the user at run-time. In order to express *Echo* as an SDF, we treated the "hslider" component, as well as the components it affected, as a constant value — the components of the signal expression involved in this are coloured grey in Figure 8.

The remaining components, which are coloured red, were then modelled as an SDF as shown in Figure 9. Using the method of modelling delay explained in Section 3.3.2, the argument provided to the delay primitive is denoted by the "N" initial tokens in the buffer between actor 1 and 4. Assuming a sampling rate (the value produced by 'fSamplingFreq' in Figure 8) of 48kHz, and observing that the "hslider" component has a range of 0.0 to 1000.0 with

a step size of 0.1, we found that "N" could take on values within the range of 0 to 47999.

4.1. Throughput

The throughput of an SDF is a measure of the rate at which it is able to process data — naturally, as actors consume and produce data, throughput is associated with the rate of actor execution in an SDFG. The period of an SDFG is the reciprocal of its throughput. It provides a measure of time elapsed between executions. There are several approaches to compute the throughput of an SDF [24]; one of which is to construct a schedule, which refers to an ordered sequence of actor firings. In particular, constructing a self-timed schedule, in which actors fire as soon as the required amount of data are available on each of their input channels, is one method to compute the maximum throughput of an SDF [25]. Once a periodic pattern of actor firings is established, the throughput can be computed as the average rate at which actors are fired in this pattern.

Naturally, existing throughput computation techniques assume that the SDFG accurately models the targeted hardware. For example, in the context of FPGA devices, where we can guarantee execution time in clock cycles for each arithmetic operator, Figure 10 shows the result of a self-timed schedule of Echo with a delay of $N = 3$ samples. It has a period of 14 cycles and, assuming a clock frequency of 400MHz, we can thus infer that the SDF in Figure 9 is able to operate at a frequency of approximately 28.5MHz; which is well above the required sampling rate of 48kHz.

4.2. Latency

Latency refers to the time difference between the arrival of a unit of data and the production of the corresponding output data [26]. Computing the latency of the SDFG is to determine the maximum time taken between an input sample and an output sample produced by the program. This can help the developer to determine if the latency of their current implementation meets their requirements. Furthermore, the process provides them with the actors whose execution times contribute to the latency of the program — reducing the execution times of these actors would therefore allow them to reduce the latency of their program.

For example from the Figure 10, we can see that there is a 20 cycle latency between the first execution of the input actor, and the first execution of the output actor, this is known as the First-to-First latency [27]. As noted in Section 1, the latency of an audio processing system is determined by factors beyond the time taken to compute a single sample. Nonetheless, this metric would be useful in guiding users towards selecting an appropriate buffer size.

5. LIMITATIONS

In this section, we discuss what we see as key limitations of our proposed approach and, where possible, suggest solutions to overcoming these limitations.

5.1. Timing Accuracy

In order to model a Faust program as an SDF, the execution times of each operator needs to be defined. However, these execution times are dependent on operation implementation which is, in turn, influenced by the hardware that it is being run on. Furthermore, operator implementations do not always have guaranteed execution times; execution times can be affected by caching effects, for example. In cases such as this, it is necessary to define operator execution times as the WCET of the given operator implementation. For example, there has been prior work in implementing schedulers for SDFGs that take into account cache architectures in embedded DSP and uniprocessor architectures that might address this issue [28, 23]. This can lead to a considerable amount of effort to accurately compute the execution times to be used in the SDF. Furthermore, the variance of the execution times of operators could impact the accuracy of the performance analysis results.

5.2. Dynamic Variables

Faust programs can include components that users can interact with at run-time to change the values of variables. If these components are used to define the length of a delay operator, it would require the buffer used to model the delay operator to have a dynamic buffer size, thereby making the program incompatible with the SDF model. While we did present a workaround for the purpose of static performance analysis in Section 4.1, it should be noted that this could incur high costs in modelling programs with dynamic buffer sizes. The workaround presented in Section 4.1, for example, would require up to 48000 different versions of SDFGs to fully represent the different states of the system. That said, there are other potential solutions.

Scenario-Aware dataflow graphs [29], which are a parametric extension of SDF that can describe multiple modes of execution for the same application, is one potential solution. This could form the topic of a future work that aims to tackle the issues with expressing dynamic variables from Faust within the dataflow model.

5.3. Full-System Analysis

The analysis performed in previous SDFG examples, such as the maximum throughput or the latency analysis, do not consider the entire system targeted, but solely the compute part of it. In addition, there are multiple layers of communication and processing between the audio input interfaces and output ones, that are not considered and limit the performance of the overall system. Nonetheless, these analysis results can ensure by how much the compute part of this application will limit the overall system.

6. RELATED WORK

Our work look into the potential benefits of dataflow modelling to perform static performance analysis on DSP programs written in Faust. Static analysis is used to extract information from the source code of a program or a binary without executing the program. It can be used to identify properties, such as correctness, to optimize, or predict the performance of programs [9].

In the case of general purpose languages such as C, data flow frequency analysis [30] has been used to predict the number of times instructions are executed. When combined with cache analysis techniques [31], this strategy could estimate hardware dependent performance of program without executing this program on the target hardware. Unfortunately, the complexity of hardware and the expressiveness of general purpose languages can often lead to situations where no pertinent estimation can be made due to cache effects and undecideable program properties.

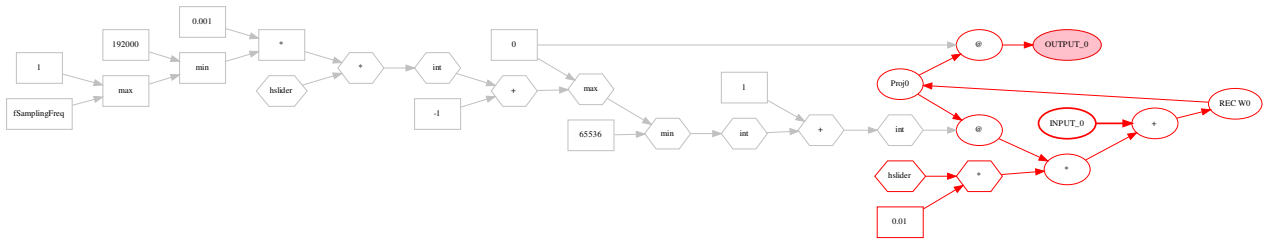


Figure 8: Signal expression of *Echo*. The actors in grey denote that they produce the value that determines the delay of *Echo* in samples.

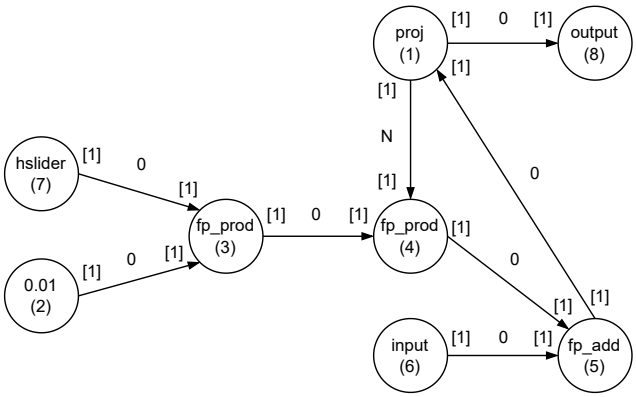


Figure 9: Subset of SDF generated from signal expression of *Echo*. The actors are labelled with their IDs in parenthesis. The edge between (1) and (4) denotes an N sample delay, which is set by the user.

In the context of DSP applications, however, where applications tend to be more regular and less dynamic, static analysis becomes much more efficient. This is especially true when programs are defined using domain specific languages such as Faust. For example, the StreamIt language [32] and the SigmaC langage [33] were used to express programs as SDF. From this intermediate representation, the compiler toolchain can generate binary code optimized for many-core hardware. In this context, static analysis is used to determine memory requirements of applications for given performance constraints or to predict performance of a particular task-to-core mapping of the application.

There are two main use cases where dataflow modeling could be used for performance analysis. First, similarly to StreamIt [32] or SigmaC [33], the application's behaviour can be faithfully expressed using dataflow models. For example, in the context of DSP hardware design, a fine-grain and functionally equivalent representation of a program is produced [34, 35, 21]. In this case, dataflow can be used to verify correctness, analyse the performance of a program, and even generate optimised code for a particular hardware target such as FPGA [34, 21] or GPU [35]. When the programming language does not exactly match dataflow models, static analysis can still be used to estimate performance and to support the compilers making decisions. This is for example what is done by the *MAPS* [36] framework, that describes applications using the C for process networks (CPN) language, and guides task-to-core mapping decision using trace-based analysis tools.

Similarly to task-to-core mapping problems, our objective is to use SDF models to help the user make decisions in choosing the right hardware, by predicting the performance for specific targets such as FPGA devices (e.g. Syfala [37]).

7. CONCLUSION

In this work, we introduce a methodology to model Faust application as SDFGs, forming the foundation for a longer term project. In the future, we plan to use this transformation to enable the use of static analysis technique for SDFGs in order to predict performance metrics such as throughput and delay. As a motivational example, we present performance estimates for the *Echo* application, by considering what could be a potential FPGA implementation.

Modelling Faust programs as SDFGs, and the associated performance analysis methods could also support the code generation process. The SDF model can act as an Intermediate Representation wherein the performance of the implementation can be evaluated and optimized prior to code generation.

Beyond the use of dataflow models to predict performance or event generate programs, an additional use case could be to provide feedback on the implementation of the Faust program, directly to the user. For example, it might be possible to provide real-time feedback to users while they are writing Faust programs in an IDE by periodically compiling and modelling their program as an SDF.

Moving forward, we advocate for integrating more high-level analysis in the programming workflow so that the user can have a comprehensive diagnosis of which parts of their program could be a performance bottleneck.

8. REFERENCES

[1] Ernest Edmonds and Sandra Pauletto, "Audiovisual discourse in digital art," in *ACM SIGGRAPH Art Gallery*, New York, NY, USA, 2004, p. 116–119.

[2] "Audiovisual art began, 10 pioneering works," `https://www.electronicbeats.net/audiovisual-art-began-10-pioneering-works/`, 2004.

[3] Romain Michon, Yann Orlarey, Stéphane Letz, Dominique Fober, and Dirk Roosenburg, "Embedded Real-Time Audio Signal Processing With Faust," *International Faust Conference (IFC-20)*, pp. 1–8, 2020.

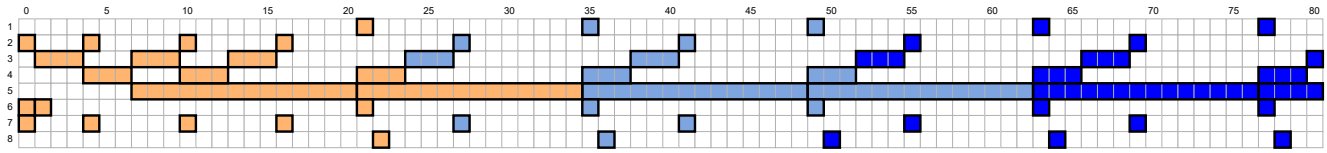


Figure 10: Self-timed schedule of *Echo* with delay of 3 samples. Initial firings are coloured *orange*, while periodic firings are coloured *blue*. The column labels represent time units while the row labels indicate the ID of the actor (as listed in Figure 9 that the firing corresponds to.

[4] David Wessel and Matthew Wright, "Problems and Prospects for Intimate Musical Control of Computers," *Computer Music Journal*, vol. 26, no. 3, pp. 11–22, sep 2002.

[5] Teemu Mäki-Patola and Perttu Hämäläinen, "Latency tolerance for gesture controlled continuous sound instrument without tactile feedback," in *ICMC*, 2004.

[6] Robert H Jack, Tony Stockman, and Andrew McPherson, "Effect of latency on performer interaction and subjective quality assessment of a digital musical instrument," in *Proceedings of the Audio Mostly Conference*, 2016.

[7] Yann Orlarey, Stéphane Letz, and Dominique Fober, "Adding automatic parallelization to faust," in *Linux Audio Conference*, 2009.

[8] Yann Orlarey, "Faust - programming language for audio applications and plugins," `https://github.com/grame-cncm/faust`, 2019.

[9] Kewen Meng and Boyana Norris, "Mira: A framework for static performance analysis," in *International Conference on Cluster Computing, CLUSTER*. 2017, pp. 103–113, IEEE.

[10] Edward A. Lee and David G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, 1987.

[11] S. Stuijk, M.C.W. Geilen, and T. Basten, "SDF^3: SDF For Free," in *Application of Concurrency to System Design, 6th International Conference, ACSD 2006, Proceedings*. June 2006, pp. 276–278, IEEE Computer Society Press, Los Alamitos, CA, USA.

[12] Maxime Pelcat, Karol Desnos, Julien Heulot, Clément Guy, Jean-François Nezan, and Slaheddine Aridhi, "Preesm: A dataflow-based rapid prototyping framework for simplifying multicore dsp programming," in *2014 6th European Embedded Design in Education and Research Conference (EDERC)*, 2014, pp. 36–40.

[13] Yann Orlarey, Dominique Fober, and Stephane Letz, "Syntactical and semantical aspects of faust," *Soft Computing*, vol. 8, no. 9, pp. 623–632, 2004.

[14] Roger B Dannenberg, "Languages for computer music," *Frontiers in Digital Humanities*, vol. 5, pp. 26, 2018.

[15] Grame Centre National de Création Musicale, "Faust news," `https://faust.grame.fr/community/news/`, Dec 2017, Accessed: 2022-03-25.

[16] Wesley M. Johnston, J. R.Paul Hanna, and Richard J. Millar, "Advances in dataflow programming languages," in *ACM Computing Surveys*, 2004, pp. 1–34.

[17] Walid A. Najjar, Edward A. Lee, and Guang R. Gao, "Advances in the dataflow computational model," *Parallel Computing*, vol. 25, no. 13, pp. 1907–1929, 1999.

[18] Maxime Pelcat, Slaheddine Aridhi, Jonathan Piat, and Jean-François Nezan, *Physical layer multi-core prototyping: A dataflow-based approach for LTE eNodeB*, Springer, 2013.

[19] Renjie Xie, Heikki Huttunen, Shuoxin Liny, Shuvra S. Bhattacharyya, and Jarmo Takala, "Resource-constrained implementation and optimization of a deep neural network for vehicle classification," in *European Signal Processing Conference*, 2016.

[20] Andrei Hagiescu, Huynh Phung Huynh, Weng Fai Wong, and Rick Siow Mong Goh, "Automated architecture-aware mapping of streaming applications onto GPUs," in *25th IEEE International Parallel and Distributed Processing Symposium*, 2011, pp. 467–478.

[21] Robert Stewart, Bernard Berthomieu, Paulo Garcia, Idris Ibrahim, Greg Michaelson, and Andrew Wallace, "Verifying parallel dataflow transformations with model checking and its application to FPGAs," *Journal of Systems Architecture*, 2019.

[22] Florent De Dinechin and Bogdan Pasca, "Designing custom arithmetic data paths with FloPoCo," *IEEE Design and Test of Computers*, vol. 28, no. 4, pp. 18–27, 2011.

[23] Kunal Agrawal, Jeremy T. Fineman, Jordan Krage, Charles E. Leiserson, and Sivan Toledo, "Cache-conscious scheduling of streaming applications," in *Annual ACM Symposium on Parallelism in Algorithms and Architectures*, 2012, pp. 236–245.

[24] Robert de Groote, "Throughput analysis of dataflow graphs," in *Handbook of Signal Processing Systems*, vol. 66, pp. 751–786. Springer, Berlin, Heidelberg, 2018.

[25] A. H. Ghamarian, M. C.W. Geilen, S. Stuijk, T. Basten, A. J.M. Moonen, M. J.G. Bekooij, B. D. Theelen, and M. R. Mousavi, "Throughput analysis of synchronous data flow graphs," in *Proceedings - International Conference on Application of Concurrency to System Design, ACSD*, 2006, pp. 25–34.

[26] A H Ghamarian, S Stuijk, T Basten, M. C.W. Geilen, and B D Theelen, "Latency minimization for synchronous data flow graphs," in *Proceedings - 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, DSD 2007*, 2007, pp. 157–164.

[27] Nico Feiertag, Kai Richter, Johan Nordlander, and Jan Jonsson, "A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics," *IEEE Real-Time System Symposium - Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*, 2008.

[28] Sanjeev Kohli, *Cache aware scheduling for synchronous dataflow programs*, Electronics Research Laboratory, College of Engineering, University of . . . , 2004.

[29] Sander Stuijk, Marc Geilen, Bart Theelen, and Twan Basten, "Scenario-aware dataflow: Modeling, analysis and implementation of dynamic applications," in *2011 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, 2011, pp. 404–411.

[30] G. Ramalingam, "Data flow frequency analysis," *SIGPLAN Notices (ACM Special Interest Group on Programming Languages)*, vol. 31, no. 5, pp. 267–275, may 1996.

[31] Johann Blieberger, Thomas Fahringer, and Bernhard Scholz, "Symbolic cache analysis for real-time systems," *Real-Time Systems*, vol. 18, no. 2, pp. 181–215, 2000.

[32] William Thies, Michal Karczmarek, and Saman Amarasinghe, "StreamIt: A language for streaming applications," in *Lecture Notes in Computer Science*. 2002, vol. 2304, pp. 179–196, Springer, Berlin, Heidelberg.

[33] Thierry Goubier, Renaud Sirdey, Stéphane Louise, and Vincent David, "ΣC: A programming model and language for embedded manycores," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2011, vol. 7016 LNCS, pp. 385–394, Springer, Berlin, Heidelberg.

[34] Mohamed A. Bamakhrama and Todor P. Stefanov, "On the hard-real-time scheduling of embedded streaming applications," *Design Automation for Embedded Systems*, vol. 17, no. 2, pp. 221–249, jun 2013.

[35] Lars Schor, Andreas Tretter, Tobias Scherer, and Lothar Thiele, "Exploiting the parallelism of heterogeneous systems using dataflow graphs on top of opencl," in *The 11th IEEE Symposium on Embedded Systems for Real-time Multimedia*, 2013, pp. 41–50.

[36] Rainer Leupers, Miguel Angel Aguilar, Juan Fernando Eusse, Jeronimo Castrillon, and Weihua Sheng, *MAPS: A Software Development Environment for Embedded Multicore Applications*, pp. 917–949, Springer Netherlands, Dordrecht, 2017.

[37] Tanguy Risset, Romain Michon, Yann Orlarey, Stéphane Letz, Gero Müller, and Adeyemi Gbadamosi, "Faust2FPGA for Ultra-Low Audio Latency: Preliminary work in the Syfala project," in *IFC2020-Internationnal Faust Conference*, dec 2020.

FAUST AND REACTIVE FUNCTIONS: A TRACED PROP OF SIGNALS AND SIGNAL RELATIONS

Nicholas Connell *

nicholas.a.connell@bath.edu

ABSTRACT

I propose a categorical interpretation of the algebra presented by Orlarey et al in their paper An Algebra For Block Diagrams [1]. The category in question is the Traced Prop of Relations on Signals. Where Sequential and Parallel composition are relational composition and monoidal product respectively, and Recursive composition is a combination of these, the Trace and more. In this interpretation reactive functions correspond to signal processors. I prove the theorem that the trace of any delayed reactive function is a reactive function. This makes explicit the need for an implicit delay in the definition of Recursion. Furthermore, I show, by way of preserving reactivity and functionality, that each of the five main FAUST operators returns a valid signal processor when fed two of them.

1. INTRODUCTION

FAUST is a functional programming language designed for digital signal processing and ensuring the longevity of digital music tools. Often functional languages are built on solid mathematical foundations. FAUST is no exception, Orlarey et al provide a denotational semantics in their paper An Algebra For Block Diagrams [1]. In this paper I aim to give an account of FAUST by providing a correspondence between it and traced monoidal categories. I will use this to show explicitly why the recursion operator is well-defined and a formal justification for the use of a 1-sample delay. Furthermore, I will use the correspondence to provide atomic decompositions of the five FAUST operators Sequential, Parallel, Merge, Split and Recursion.

Recently there has been considerable interest in algebraic and graphical accounts of systems, like the work of Orlarey et al. Especially in the literature focused on String Diagrams: Joyal et al introduced traced monoidal categories [2], Bonchi et al applied this to create the resource calculus, a string diagrammatic language, for use in concurrency theory [3], and in Picturing Quantum Processes Coecke et al use string diagrams to teach quantum theory [4].

I will use a categorical interpretation to find interesting and rigorous mathematical answers to questions you might not have thought to ask. Such as, whether the five basic composition operators in FAUST will always return a valid signal processor when given two such processors. Questions like these should be investigated exactly because they have seemingly obvious answers. They test the foundations of FAUST and their answers provide a deeper understanding which might lead to innovation and discovery in the future.

To answer questions such as these, one needs a model for signals, signal processors and processor composition operators. The model I put forward is the Traced Prop of Relations on Signals (Sections 3 and 4). Where objects are tuples of signals and morphisms are relations on signal tuples. This puts the emphasis on how computation happens instead of what is being computed. Furthermore, composition and monoidal product are the analogues of the Sequential and Parallel operators respectively. The Merge and Split operators can be built from these with a few primitive processors. The Recursion operator can be constructed with the trace, composition, monoidal product and some more primitives. The mathematical analogue for signal processors will be reactive functions, a subset of the signal relations. In the literature these also go by the name causal functions ([5]).

We begin with a prop $\mathbb{S}$ of relations on tuples of signals. This provides structure to support the Sequential, Parallel, Merge, Split and Recursion operators. FAUST programs are deterministic, so relations are not a suitable vehicle to model them. Instead, we will employ reactive functions, which always give a certain output for a fixed input, depend on past input and do not predict future input. These are indeed morphisms in our prop. We will show that they are closed under all operators except for feedback; decomposing the operators into more atomic elements allows us to show this. Feedback, in the sense of a trace on a monoidal category, unfortunately has the potential to return something that is not a function. But, we can recover closedness by the familiar inclusion of a 1-sample delay; this is the remit of Theorem 1.

The main contributions of this paper are the proof of Theorem 1: The trace of any delayed reactive function is a reactive function, that all five of FAUST's operators are closed with respect to reactive functions and the algebraic decompositions of these operators.

To begin with I will provide some preliminary information in Section 2. Such as, explanations for the notation you will see and necessary definitions for signals, processors and the five main composition operators in FAUST. I have used definitions from An Algebra for Block Diagrams, and from the FAUST documentation website for the signals and composition operators. And I have tried to use the same notation where it is possible and does not lead to confusion. [1], [6] Lastly, Section 7 provides some concluding remarks.

2. PRELIMINARIES

The first step in developing a mathematical model is to gain a detailed understanding of what is being modelled. This section therefore is devoted to a basic understanding of FAUST via preliminary definitions for signals, signal processors and it's five composition operators. Much of the information gathered here is derived from the Faust Documentation and An Algebra for Block Diagrams by Orlarey et al. [6], [1]

2.1. Signals and Signal Processors

Unless otherwise stated, in this paper $\mathbb{N} = \mathbb{N} \cup \{0\}$.

* Dissertation Master of Computing in Computer Science and Mathematics at the University of Bath

Signals are discrete functions of time. $\mathbb{S}$ is the set of all signals, a signal tuple of size m is an element of the cartesian product $\mathbb{S}^m$.

$$\mathbb{S} := \{s : \mathbb{N} \to \mathbb{R}\}.$$

Equivalently, one could use $\mathbb{Z}$ as the domain and state that $s(t) = 0 \ \forall \ t < 0$. This would allow a non-recursive definition of delay, but would entail dealing with an 'infinite history' of zeros. One could also use sequence notation $s = (s_n)_{n\in\mathbb{N}}$. However, I will stick with the uncomplicated version provided by Orlarey et al. In a diagram the indexing of inputs and outputs will start at 1 on the lowest signal count up to the highest one.

Signal Processors are functions on signal tuples. $\mathbb{P}$ denotes the set of all signal processors.

$$\mathbb{P} := \bigcup_{n,m\geq 0} \{p : \mathbb{S}^m \to \mathbb{S}^n\}.$$

IO$ins(p)$ and $outs(p)$ return the number of inputs or outputs of a given signal processor, below is a formal definition.

$$ins() : \mathbb{P} \to \mathbb{N} : ins(p : \mathbb{S}^m \to \mathbb{S}^n) = m$$
$$outs() : \mathbb{P} \to \mathbb{N} : outs(p : \mathbb{S}^m \to \mathbb{S}^n) = n$$

2.2. Faust Operators

Sequential Composition: (fig. 1) for compatible processors B and C, `B:C` will link the outputs of B with the inputs C. B's first output will supply C's first input, etc. B and C are compatible if $outs(B) = ins(C)$.

Merge Composition: (fig. 2) for compatible processors B and C, `B:>C` will sum collections of B's outputs and pass the results to C's inputs. B and C are compatible if $outs(B) = k \times ins(C)$. Outputs b and β of B are in the same collection if $b \equiv \beta \bmod ins(C)$. The sum of a collection with representative b is passed to input c of C if $c \equiv b \bmod ins(C)$.

Split Composition: (fig. 3) for compatible processors B and C, `B<:C` will copy a given output of B to possibly many inputs of C. B and C are compatible if $k \times outs(B) = ins(C)$. An output b of B supplies an input c of C if $c \bmod outs(B) \equiv b$.

Parallel Composition: (fig. 4) for any two processors, `B,C` stacks C on top of B. The resulting processor A will have $ins(A) = ins(C) + ins(B)$, where an input i of A corresponds to input i of C if $i \leq ins(C)$ or to input $j = i - ins(C)$ of B otherwise. The same goes for A's outputs.

Recursive Composition: (fig. 5) for compatible processors B and C, `B~C` creates processor A based on a feedback relationship. The first $ins(C)$ outputs of B are duplicated. One copy supplies the inputs of C, whose outputs first pass through a one-sample delay before supplying the first $outs(C)$ inputs of B. The other copy is merged with B's unduplicated outputs to become the outputs of A. B and C are compatible if $ins(B) \geq outs(C)$ and $outs(B) \geq ins(C)$. The inputs of A are the last $ins(B) - outs(C)$ inputs of B.

As you can see, each composition operator has their own compatibility rules. In particular, any two processors may be composed in parallel, but you may only compose two processors in sequence if the output of one matches the input of another. These are exactly the requirements for combining morphisms by monoidal product and by composition in monoidal categories. This is explored further in Section 3.

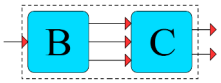


Figure 1: *Sequential Composition. [1]*

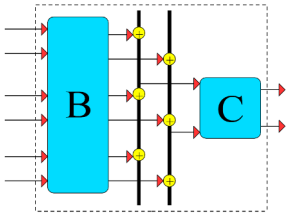


Figure 2: *Merge Composition. [1]*

Furthermore, there is an obvious hierarchy of complexity. We have Parallel and then Sequential with the two least complex descriptions. Next we have Split and Merge, which seem to be two different ways of generalising Sequential composition. Consider their notations `:` `<:` `:>` and what happens when we set $k = 1$. This suggests that Split and Merge are not atomic and could be decomposed into less complex entities. This is explored further in Section 3.4. Lastly, Recursion is the most complex by far. In fact, recursion seems to behave like a trace in a traced category. This is explored in Section 6.

2.3. Useful Functions

I have collected the following functions here because they are used throughout the paper.

Gluing is the parallel composition of signal tuples. In a diagram it aligns the signals vertically putting the first argument below the second. For $\mathbf{x} \in \mathbb{S}^m$, $\mathbf{y} \in \mathbb{S}^n$, we define this index-wise

$$||: (\mathbb{S}^m \times \mathbb{S}^n) \to \mathbb{S}^{m+n}$$
$$(\mathbf{x}||\mathbf{y})_i = \begin{cases} \mathbf{y}_{i-m}, & \text{for } i > m \\ \mathbf{x}_i, & \text{for } i \leq m. \end{cases}$$

I chose to use $||$ for parallel alignment instead of a comma as in FAUST to distinguish many signals in parallel from consecutive elements within a signal $s = (s(0), s(1), ...)$.

Restrict takes a signal tuple and a time-step as input and cuts off all the values of the tuple after the time-step. Let $n \in \mathbb{N}$ and $\mathbf{x} = (\mathbf{x}(0), \mathbf{x}(1), ...) \in \mathbb{S}^m$

$$\upharpoonright n : \mathbb{S}^m \to \mathbb{R}^{m\times n},$$
$$\mathbf{x} \upharpoonright n = (\mathbf{x}(0), ..., \mathbf{x}(n)).$$

That is $\mathbf{x} \upharpoonright n$ gives the first $n+1$ m-tuples of values of the signal $\mathbf{x}$. For example, let the signal $x = 0, 1, 2, ...$ then $x \upharpoonright 1 = 0, 1$.

Restrict obviously distributes over Gluing. That is, for $m, n \in \mathbb{N}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{S}^m$ we have

$$(\mathbf{x}||\mathbf{y}) \upharpoonright n = (\mathbf{x} \upharpoonright n)||(\mathbf{y} \upharpoonright n).$$

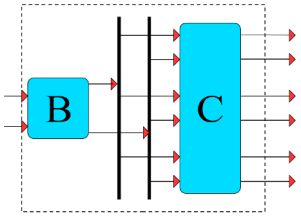


Figure 3: *Split Composition. [1]*

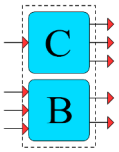


Figure 4: *Parallel Composition. [1]*

Restrict also lets us define partial evaluation for a signal processor. Let $k, m, n \in \mathbb{N}$ then the partial evaluation of a signal $\mathbf{x} \in \mathbb{S}^m$ at time n by a processor $p : \mathbb{S}^m \to \mathbb{S}^k$ is

$$p(\mathbf{x} \upharpoonright n) := p(\mathbf{x}) \upharpoonright n.$$

Slicing takes a signal tuple, and start and end indices as input. It cuts off the signals in the tuple that are above or below the provided indices. Let $a, b, m \in \mathbb{N}$ such that $a \leq b \leq m$ and $\mathbf{x} \in \mathbb{S}^m$, then

$$[a : b] : \mathbb{S}^m \to \mathbb{S}^{b-a+1}$$
$$\mathbf{x}[a : b] = \mathbf{x}_a \| ... \| \mathbf{x}_b.$$

Slicing and Restriction too may be composed in either order. That is for $a, b, m, n \in \mathbb{N}$ and $\mathbf{x} \in \mathbb{S}^m$ we have

$$(\mathbf{x} \upharpoonright n)[a : b] = (\mathbf{x}[a : b]) \upharpoonright n.$$

Delay is defined inductively as follows. For all $n \in \mathbb{N}$

$$\Delta_{\mathbb{S}^m}(\mathbf{x}) \upharpoonright n = \begin{cases} 0, \text{ for } n = 0 \\ 0, (\mathbf{x} \upharpoonright n - 1), \text{ for } n > 0. \end{cases}$$

In other words, the delay shifts the time-step of all the values of $\mathbf{x}$ up by one and then inserts a 0 at beginning. For example, let $x \in \mathbb{S}$ be the constant stream of 1s, then $\Delta_{\mathbb{S}}(x) = 011111...$.

2.4. Reactivity

Faust programs process audio signals in real-time, and so the model should reflect this. A signal processor cannot predict the future, thus its output may only be based on what it has already seen. As a consequence, if a signal processor is given two inputs that start the same but are ultimately different, the two outputs should be identical at least until the time when the inputs first differ in value. This notion is formalised below:

A processor $p : \mathbb{S}^m \to \mathbb{S}^k$ is **reactive** if, for $\mathbf{x}, \mathbf{y} \in \mathbb{S}^m$ satisfying $\mathbf{x} \upharpoonright n = \mathbf{y} \upharpoonright n$, for some $n \in \mathbb{N}$, we have

$$p(\mathbf{x}) \upharpoonright n = p(\mathbf{y}) \upharpoonright n.$$

Here I present a quick example of how to prove that signal processors are reactive.

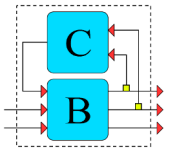


Figure 5: *Recursive Composition. [1]*

Lemma 1. *$\Delta_{\mathbb{S}^m} : \mathbb{S}^m \to \mathbb{S}^m$ and $id_{\mathbb{S}^m} : \mathbb{S}^m \to \mathbb{S}^m$ are reactive functions for any $m \in \mathbb{N}$.*

Proof. Suppose $\mathbf{x}, \mathbf{y} \in \mathbb{S}^m$ and $\mathbf{x} \upharpoonright n = \mathbf{y} \upharpoonright n$ for some $n \in \mathbb{N}$. Then

$$id_{\mathbb{S}^m}(\mathbf{x}) \upharpoonright n = \mathbf{x} \upharpoonright n = \mathbf{y} \upharpoonright n = id_{\mathbb{S}^m}(\mathbf{y}) \upharpoonright n.$$

$$\begin{aligned} \Delta_{\mathbb{S}^m}(\mathbf{x}) \upharpoonright n &= 0, \mathbf{x} \upharpoonright (n-1) = 0, (\mathbf{x} \upharpoonright n) \upharpoonright (n-1) \\ &= 0, (\mathbf{y} \upharpoonright n) \upharpoonright (n-1) = 0, \mathbf{y} \upharpoonright (n-1) \\ &= \Delta_{\mathbb{S}^m}(\mathbf{y}) \upharpoonright n. \end{aligned}$$

As m and n were arbitrary, the lemma is proven. □

3. THE PROP OF RELATIONS ON SIGNALS

A Prop is a strict symmetric monoidal category, where we can define a bijection between its objects and the natural numbers and the action of the monoidal product on objects is like addition for natural numbers.

3.1. Definition of $Rel_{\mathbb{S}}$

Let $Rel_{\mathbb{S}}$ the Prop of relations on signal tuples be defined as follows. I begin with the structure of the underlying category, this includes mathematical analogues for the signal processors, their inputs and outputs and composing them in sequence. I have defined this Prop in terms of relations and not just reactive functions. This is fine because reactive functions are relations, so what holds for relations holds for reactive functions too.

- **Objects** are the natural numbers, each $k \in \mathbb{N}$ corresponds to its own set $\mathbb{S}^k$ of all possible tuples of signals of size k.
- **Morphisms** $R : \mathbb{S}^k \to \mathbb{S}^l$ are relations in the set $Rel_{\mathbb{S}}(\mathbb{S}^k, \mathbb{S}^l) = \{R \subseteq \mathbb{S}^k \times \mathbb{S}^l\}$.
- **Identity Morphisms** $id_k : \mathbb{S}^k \to \mathbb{S}^k$ are just identity relations.
- **Composition of Morphisms** in $Rel_{\mathbb{S}}$ is just composition of relations, for $\mathbf{x} \in \mathbb{S}^k$ and $\mathbf{y} \in \mathbb{S}^m$, $f : \mathbb{S}^k \to \mathbb{S}^l$ and $g : \mathbb{S}^l \to \mathbb{S}^m$ we have $(\mathbf{x}, \mathbf{y}) \in g \circ f$ iff $g(f(\mathbf{x})) = \mathbf{y}$.

Now I add the structure necessary for a monoidal category. This provides a mathematical analogue for composing in parallel.

- **Monoidal Product on Objects** is defined by addition, $\mathbb{S}^k \otimes \mathbb{S}^l = \mathbb{S}^{k+l}$.
- The **Identity Object** is the singleton set containing the empty tuple $\mathbb{S}^0$.
- **Monoidal Product on Morphisms** stacks them in parallel, for $(\mathbf{x}, f(\mathbf{x})) \in f : \mathbb{S}^k \to \mathbb{S}^l$ and $(\mathbf{y}, g(\mathbf{y})) \in g : \mathbb{S}^m \to \mathbb{S}^n$ we have $(\mathbf{x}||\mathbf{y}, f(\mathbf{x})||g(\mathbf{y})) \in f \otimes g : \mathbb{S}^{k+m} \to \mathbb{S}^{l+n}$.

- The **Associator** for $\mathbb{S}^k$, $\mathbb{S}^l$ and $\mathbb{S}^m$ is the identity on $\mathbb{S}^{k+l+m}$, $a_{k+l+m} : \mathbb{S}^{k+l} \otimes \mathbb{S}^m \to \mathbb{S}^k \otimes \mathbb{S}^{l+m} = id_{k+l+m}$
- The **Left and Right Unitors** are identities, $l_k : \mathbb{S}^{0+k} \to \mathbb{S}^k = id_k = r_k : \mathbb{S}^{k+0} \to \mathbb{S}^k$.

This last structural component adds the ability to swap and change the order of the input or output signals.

- The **Symmetry** for $\mathbb{S}^k$ and $\mathbb{S}^l$ are defined as $c_{k,l} : \mathbb{S}^{k+l} \to \mathbb{S}^{l+k}$ where $c_{k,l}(\mathbf{x}||\mathbf{y}) = \mathbf{y}||\mathbf{x}$.

I will from now on refer to $\mathbb{S}^k$ as k and $\mathbb{S}^{k+l}$ as $k + l$.

3.2. Verifying Axioms of Symmetric Strict Monoidal Categories

The axioms for a category are satisfied. Namely that composition is associative. The definition we have chosen for $Rel_{\mathbb{S}}$ means it is a subcategory of **REL** the category of sets and relations and thus a category in its own right.

3.2.1. Monoidal Structure

According to Hasegawa in [7] the monoidal product of a monoidal category must be a bifunctor $\otimes : Rel_{\mathbb{S}} \times Rel_{\mathbb{S}} \to Rel_{\mathbb{S}}$. That is $\otimes$ must preserve identity morphisms and composition of morphisms:

$$id_k \otimes id_l = id_{k+l} \tag{1}$$
$$(f_2 \circ f_1) \otimes (g_2 \circ g_1) = (f_2 \otimes g_2) \circ (f_1 \otimes g_1). \tag{2}$$

I claim the monoidal product satisfies both the equations. To verify the first equation consider $\mathbf{x}_i \in k$ and $\mathbf{y}_i \in l$ for $i = 1, 2$.

$$\begin{aligned}(\mathbf{x}_1||\mathbf{y}_1, \mathbf{x}_2||\mathbf{y}_2) \in id_k \otimes id_l &\implies (\mathbf{x}_1, \mathbf{x}_2) \in id_k, (\mathbf{y}_1, \mathbf{y}_2) \in id_l \\ &\implies \mathbf{x}_1 = \mathbf{x}_2,\ \mathbf{y}_1 = \mathbf{y}_2.\end{aligned}$$

The first implication comes from the definition of $\otimes$, the second from the definition of identities. By the chain of implications above we can conclude that $id_k \otimes id_l \subseteq id_{k+l}$. For the opposite inclusion consider $(\mathbf{x}, \mathbf{x}) \in id_{k+l}$. Notice that there exists $\mathbf{x}_1 \in k$ and $\mathbf{x}_2 \in l$ such that $\mathbf{x}_1||\mathbf{x}_2 = \mathbf{x}$, this leads to the following implication

$$\begin{aligned}&(\mathbf{x}_1, \mathbf{x}_1) \in id_k, (\mathbf{x}_2, \mathbf{x}_2) \in id_l \implies \\ &(\mathbf{x}, \mathbf{x}) = (\mathbf{x}_1||\mathbf{x}_2, \mathbf{x}_1||\mathbf{x}_2) \in id_k \otimes id_l.\end{aligned}$$

Therefore $id_{k+l} \subseteq id_k \otimes id_l$. And so we get the equality required.

To prove that the second equation holds consider the following logic. Let $f_1 : k_1 \to l_1$, $f_2 : l_1 \to m_1$, $g_1 : k_2 \to l_2$ and $g_2 : l_2 \to m_2$. Let $\mathbf{x}_1 \in k_1$, $\mathbf{x}_2 \in m_1$, $\mathbf{y}_1 \in k_2$ and $\mathbf{y}_2 \in m_2$. Then

$$\begin{aligned}&(\mathbf{x}_1||\mathbf{y}_1, \mathbf{x}_2||\mathbf{y}_2) \in (f_2 \circ f_1) \otimes (g_2 \circ g_1) \iff \\ &(\mathbf{x}_1, \mathbf{x}_2) \in f_2 \circ f_1, (\mathbf{x}_1, \mathbf{x}_2) \in g_2 \circ g_1 \iff \\ &(\mathbf{x}_1, f_1(\mathbf{x}_1)) \in f_1, (f_1(\mathbf{x}_1), \mathbf{x}_2) \in f_2, \\ &(\mathbf{y}_1, g_1(\mathbf{y}_1)) \in g_1, (g_1(\mathbf{y}_1), (\mathbf{y}_2)) \in g_2 \iff \\ &(\mathbf{x}_1||\mathbf{y}_1, f_1(\mathbf{x}_1)||g_1(\mathbf{y}_1)) \in f_1 \otimes g_1, \\ &(f_1(\mathbf{x}_1)||g_1(\mathbf{y}_1), \mathbf{x}_2||\mathbf{y}_2) \in f_2 \otimes g_2 \iff \\ &(\mathbf{x}_1||\mathbf{y}_1, \mathbf{x}_2||\mathbf{y}_2) \in (f_2 \otimes g_2) \circ (f_1 \otimes g_1)\end{aligned}$$

The first and third ' $\iff$ ' occur due to the definition of $\otimes$, while the second and fourth are due the definition of composition. The chain of 'if and only if' statements above reveal that the second equation in (2) does indeed hold true.

Next in [7] Hasegawa states that the associator, and left and right unitors must be natural isomorphisms. This is trivially true, because for every triplet of objects (resp. single object) the associator (resp. left and right unitors) is an identity morphism. Furthermore, the associator must satisfy the pentagon commuting diagram (fig 6) and all three must satisfy the triangle commuting diagram (fig 7). However, because they are all identities they do satisfy the commuting diagrams.

Furthermore, we can conclude that this monoidal strucure is strict, because the Associator and the Unitors are identity natural transformations.

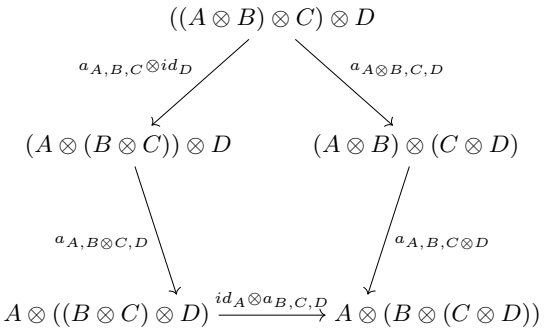


Figure 6: *Pentagon Diagram.*

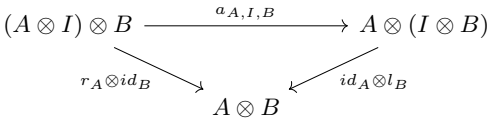


Figure 7: *Triangle Diagram.*

3.2.2. Symmetry Structure

The symmetry must satisfy four conditions. The first is that $c_{k,l}$ must be a bijection with inverse $c_{l,k}$. Consider that both of these symmetries are functions and that for $\mathbf{x} \in k$ and $\mathbf{y} \in l$ we have

$$\begin{aligned}c_{l,k}(c_{k,l}(\mathbf{x}||\mathbf{y})) &= c_{l,k}(\mathbf{y}||\mathbf{x}) = \mathbf{x}||\mathbf{y} \\ c_{k,l}(c_{l,k}(\mathbf{y}||\mathbf{x})) &= c_{k,l}(\mathbf{x}||\mathbf{y}) = \mathbf{y}||\mathbf{x}.\end{aligned}$$

That is, $c_{l,k} \circ c_{k,l} = id_{k+l}$ and $c_{k,l} \circ c_{l,k} = id_{l+k}$. As the two symmetries are inverses of one another, we can conclude that they are bijections. And so the first condition is satisfied.

The second condition is to be a natural isomorphism with respect to the functors $= \otimes -$ and $- \otimes =$. This condition holds if the symmetry satisfies the following two equations for any $f : k \to l$ and $g : m \to n$

$$\begin{aligned}(g \otimes f) \circ c_{k,m} &= c_{l,n} \circ (f \otimes g) \\ c_{n,l} \circ (g \otimes f) &= (f \otimes g) \circ c_{m,k}.\end{aligned}$$

To see that the first equation holds, let $\mathbf{x} \in k$ and $\mathbf{y} \in m$ then

$$\begin{aligned}((g \otimes f) \circ c_{k,m})(\mathbf{x}||\mathbf{y}) &= (g \otimes f)(\mathbf{y}||\mathbf{x}) \\ &= g(\mathbf{y})||f(\mathbf{x}) \\ &= c_{l,n}(f(\mathbf{x})||g(\mathbf{y})) \\ &= (c_{l,n} \circ (f \otimes g))(\mathbf{x}||\mathbf{y}).\end{aligned}$$

The proof for the second equation is similar. And so the second condition is met.

The third condition tackles coherence with the Unitors. It requires that this equation holds true

$$l_k \circ c_{k,0} = r_k.$$

To see that this holds let $\mathbf{x} \in k$ and let $e \in 0$ be the empty tuple. Recall that l_k and r_k are identities, then

$$c_{k,0}(\mathbf{x}||e) = e||\mathbf{x} = \mathbf{x} = \mathbf{x}||e$$

proves that the third condition is fulfilled.

The fourth condition tackles coherence with the Associator. It requires that the symmetry satisfies the commuting diagram in fig. 8. Because the Associator is the identity, we can simply this diagram to get fig. 9. Which represents this equation

$$(id_n \otimes c_{m,k}) \circ (c_{m,n} \otimes id_k) = c_{m,n+k}.$$

To see that this is true, let $\mathbf{x} \in m, \mathbf{y} \in n, \mathbf{z} \in k$ and consider

$$\begin{aligned}(id_n \otimes c_{m,k})((c_{m,n} \otimes id_k)(\mathbf{x}||\mathbf{y}||\mathbf{z})) &= (id_n \otimes c_{m,k})(\mathbf{y}||\mathbf{x}||\mathbf{z}) \\ &= (\mathbf{y}||\mathbf{z}||\mathbf{x}) \\ &= (c_{m,n+k})(\mathbf{x}||\mathbf{y}||\mathbf{z}).\end{aligned}$$

And so the fourth condition holds true. Thus, having met all four conditions we can conclude that the candidate symmetry defined at the start of this section is in fact a symmetry on $Rel_{\mathbb{S}}$.

$$\begin{array}{ccccc}(A \otimes B) \otimes C & \xrightarrow{a_{A,B,C}} & A \otimes (B \otimes C) & \xrightarrow{c_{A,B\otimes C}} & (B \otimes C) \otimes A \\ \downarrow{\scriptstyle c_{A,B}\otimes id_C} & & & & \downarrow{\scriptstyle a_{B,C,A}} \\ (B \otimes A) \otimes C & \xrightarrow{a_{B,A,C}} & B \otimes (A \otimes C) & \xrightarrow{id_B \otimes c_{A,C}} & B \otimes (C \otimes A)\end{array}$$

Figure 8: *Associativity Coherence.*

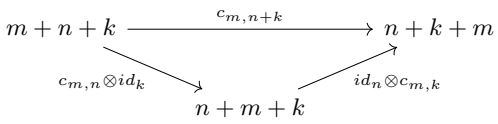


Figure 9: *Simplified Associativity Coherence.*

3.3. Monoidal Product And Sequential Composition Preserve Reactivity

Composition: Let $f : K \to L$ and $g : L \to M$ be reactive functions. $g \circ f$ is a function by way of composition preserving functionality and the fact that we're working deep within the category $\mathbf{Set}$. Now, suppose $t, s \in K$ and $n \in \mathbb{N}$ such that $t \upharpoonright n = s \upharpoonright n$. By reactivity of f we get that $f(t) \upharpoonright n = f(s) \upharpoonright n$. Moreover, by g's reactivity and the fact that $f(t), f(s) \in L$, we have $g(f(t)) \upharpoonright n = g(f(s)) \upharpoonright n$. This is exactly $g \circ f(t) \upharpoonright n = g \circ f(s) \upharpoonright n$. As t, s and n were all arbitrary we arrive at the conclusion that the composition of reactive functions is itself reactive.

Monoidal Product: Let $X_1, X_2, Y_1, Y_2 \in \mathbb{N}$ and let $f_1 : X_1 \to Y_1$ and $f_2 : X_2 \to Y_2$ be reactive functions. In the Prop of signal processors and signal tuples we can construct:

$$\begin{aligned}&f_1 \otimes f_2 : X_1 + X_2 \to Y_1 + Y_2, \text{where} \\ &(f_1 \otimes f_2)(x_1||x_2) \mapsto f_1(x_1)||f_2(x_2).\end{aligned}$$

Suppose, that we have some $n \in \mathbb{N}$ and $(x_1||x_2), (z_1||z_2) \in X_1 + X_2$ such that $(x_1||x_2) \upharpoonright n = (z_1||z_2) \upharpoonright n$, then

$$\begin{aligned}x_i \upharpoonright n = z_i \upharpoonright n &\implies f_i(x_i) \upharpoonright n = f_i(z_i) \upharpoonright n, \text{ for } i = 1, 2 \\ &\implies (f_1(x_1)||f_2(x_2)) \upharpoonright n = (f_1(z_1)||f_2(z_2)) \upharpoonright n. \\ &\implies (f_1 \otimes f_2)(x_1||x_2) \upharpoonright n = (f_1 \otimes f_2)(z_1||z_2) \upharpoonright n.\end{aligned}$$

This establishes that $(f_1 \otimes f_2)$ is reactive. Its functionality comes from the fact that $\otimes$ is closed with respect to signal processors, i.e functions.

3.4. Merge and Split Preserve Reactivity

Using $Rel_{\mathbb{S}}$ we can describe a few of Faust's composition operators. Firstly, ':' the sequential composition and ',' the parallel composition are just special cases of composition of signal relations and the monoidal product of signal relations. They are special because Faust only deals with reactive functions, which are a subset of all the relations between to signal tuple sets. In the previous section we showed that they preserved reactivity

3.4.1. Projections

These constitute a special subset of the signal processors, they are also a special case of Slice $[a : b]$ where $a = b$. We will define them in the usual way, with $M, i \in \mathbb{N}$ and $i \leq M$:

$$\begin{aligned}&\pi_i : M \to 1, \\ &\pi_i : (s_1\|...\|s_m) \mapsto s_i.\end{aligned}$$

To see that these are reactive, consider $\mathbf{x}, \mathbf{y} \in M$ such that for some $n, i \in \mathbb{N}$ $\mathbf{x} \upharpoonright n = \mathbf{y} \upharpoonright n$, then

$$\pi_i(\mathbf{x}) \upharpoonright n = x_i \upharpoonright n = y_i \upharpoonright n = \pi_i(\mathbf{y}) \upharpoonright n.$$

3.4.2. Copy

For some $k, M \in \mathbb{N}$ copy $\gamma_{k,M} : M \to kM$ takes an input from M and produces k copies of it. Note that for $k = 1$ copy is the identity. It is defined by

$$\gamma_{k,M}(\mathbf{x}) = \mathbf{x}_1\| \cdots \|\mathbf{x}_k.$$

Here $\mathbf{x} = \mathbf{x}_i$ for all $1 \leq i \leq k$, I simply enumerate the signals to illustrate the number of copies that are made. This function is obviously reactive. Let $\mathbf{x}, \mathbf{y} \in M$ such that for some $k, n, M \in \mathbb{N}$ $\mathbf{x} \upharpoonright n = \mathbf{y} \upharpoonright n$, then

$$\begin{aligned}\gamma_{k,M}(\mathbf{x}) \upharpoonright n &= (\mathbf{x}\| \cdots \|\mathbf{x}) \upharpoonright n = \mathbf{x} \upharpoonright n\| \cdots \|\mathbf{x} \upharpoonright n \\ &= \mathbf{y} \upharpoonright n\| \cdots \|\mathbf{y} \upharpoonright n = (\mathbf{y}\| \cdots \|\mathbf{y}) \upharpoonright n \\ &= \gamma_{k,M}(\mathbf{y}) \upharpoonright n\end{aligned}$$

As all the variables were arbitrary copy is a reactive function.

3.4.3. Split Composition

Recalling the definition of the '<:' Split composition from section 1, let $k, M, N, L \in \mathbb{N}$, $\mathbf{x} \in M$, $p : M \to N$ and $q : kN \to L$, then

$$\begin{aligned}
&\mathtt{p<:q} : M \to L, via\\
&\mathtt{p<:q} : \mathbf{x} \mapsto q((p_1(\mathbf{x}))\| \cdots \|(p_k(\mathbf{x})))\\
&\mathtt{p<:q} = q \circ (p_1 \otimes \cdots \otimes p_k) \circ \gamma_{k,M}.
\end{aligned}$$

In the second and third lines, I use the same enumeration for illustration technique as in (3.4.2).

3.4.4. Merge Composition

Recalling the definition of ':>' Merge composition from section 1, let $k, M, N, L \in \mathbb{N}$, $\mathbf{x} \in M$, $p : M \to kN$ and $q : N \to L$, then

$\mathtt{p:>q} : M \to L$, defined by

$$\mathtt{p:>q} : \mathbf{x} \mapsto q\left(\sum_{j=0}^{k-1} \pi_{1+jN}(p(\mathbf{x}))\middle\| \cdots \middle\|\sum_{j=0}^{k-1} \pi_{N+jN}(p(\mathbf{x}))\right)$$

$$\mathtt{p:>q} = q \circ \left(\sum_{j=0}^{k-1} \pi_{1+jN} \circ p \otimes \cdots \otimes \sum_{j=0}^{k-1} \pi_{N+jN} \circ p\right) \circ \gamma_{N,M}$$

. Note that trivially addition is a reactive function.

The algebraic versions of Split and Merge operators comprise only the sequential composition and monoidal product of reactive functions. We know these preserve reactivity and functionality. So, we can conclude that the Merge and Split operators also preserve reactivity and functionality. As we can see, these two operators are not atomic but can be broken down into more atomic parts.

We can't yet describe '$\sim$' the Recursion operator. This is because Props don't have a notion of 'feedback' or 'looping'. In the next section we look at traces on $Rel_{\mathbb{S}}$ so that we can have a mathematical description of feedback.

4. TRACE ON REL$_{\mathbb{S}}$

In this section we investigate the existence of a trace defined on the Prop of signals and relations $Rel_{\mathbb{S}}$. We begin by defining a candidate trace via its action on signal relations. Then, we proceed to provide evidence that it satisfies all the axioms required in [7] for it to be a trace: functionality, naturality and adherence to four coherence axioms. The theory of traced monoidal categories, as seen in [7], provides an algebraic account of systems with feedback. We shall now show that $Rel_{\mathbb{S}}$ has a trace, which will let us model FAUST's recursion operator.

4.1. Definition of $Tr^X_{A,B}$

The first hypothesis in Hasegawa's definition of trace is that we are working within a balanced monoidal category. Luckily for us a Prop, a strict symmetric monoidal category, is, in particular, a balanced monoidal category.

The trace is a family of functions (we'll prove this property soon). Each function relates to a triplet of objects (A, B, X) in $Rel_{\mathbb{S}}$ as map from one set of signal relations to another.

$$Tr^X_{A,B} : Rel_{\mathbb{S}}(A \otimes X, B \otimes X) \to Rel_{\mathbb{S}}(A, B),$$

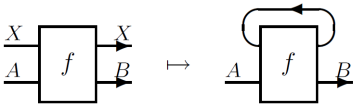


Figure 10: *Trace Action. [7]*

for $f \subseteq (A \otimes X \times B \otimes X)$

$$(a, b) \in \ Tr^X_{A,B}(f) \Leftrightarrow \exists\ x \in X \text{ s.t. } (a\|x, b\|x) \in f. \quad (3)$$

I shall refer to the above logic as the 'trace condition'.

4.2. Functionality of $Tr^X_{A,B}$

I claimed earlier that each member of this family is a function. To see that this is true, we show that each member always has an output for a given input and that this output is unique.

To see that $Tr^X_{A,B}$ always produces a relation for a given input relation, consider the set $\{(a, b) \in (A{\times}B) | \exists\ x \in X s.t. (a||x, b||x) \in (A \otimes X \times B \otimes X)\}$. This set is either infinite, finite or empty, in all cases it remains a subset of $(A \times B)$. Thus, the trace will always produce an output relation between A and B when given an input relation between $A \otimes X$ and $B \otimes X$.

Fix $X, A,$ and B. Let $f \subseteq (A \otimes X) \times (B \otimes X)$. Assume, for contradiction, that $\exists g, g' \subseteq (A \times B)$ such that $g \neq g'$ and

$$Tr^X_{A,B}(f) = g \text{ and } Tr^X_{A,B}(f) = g'.$$

Without loss of generality we can assume there is a pair $(a, b) \in (A \times B)$ such that $(a, b) \in g$, but $(a, b) \notin g'$. However, by (3)

$$\begin{aligned}
(a, b) \in g &\Leftrightarrow \exists\ x \in X \text{ s.t. } (a\|x, b\|x) \in f\\
&\Leftrightarrow (a, b) \in g',
\end{aligned}$$

which is a contradiction. Thus, the assumption that g and g' are different is incorrect. This is the equivalent to saying $Tr^X_{A,B}$ has unique outputs for given inputs.

A function is a relation has existence and uniqueness of output, we have shown that $Tr^X_{A,B}$ has both these properties and we can conclude that it's a function.

4.3. Naturality

Many definitions of trace also call for the family of functions to be 'natural'. Each member must be natural in the traced and untraced variables. This property ensures that you can compose morphisms with the input being traced and then take the trace or compose with output being traced and then take the trace and still arrive at the same end product.That is, for naturality in the trace line, we want the following equation to be true.

$$Tr^X_{A,B}((id_B \otimes h) \circ f) = Tr^X_{A,B}(f \circ (id_A \otimes h)).$$

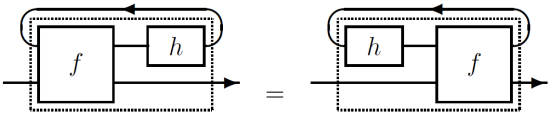


Figure 11: *Naturality. [7]*

This property is the same the sliding coherence condition for Joyal, Street and Verity's paper

To see that this is true, let $f \subseteq (A \otimes X) \times (B \otimes X)$, $h \subseteq X \times X$, id_A, id_B be the identity relations on A and B respectively, and suppose that $\exists\ a \in A, b \in B$ such that

$$(a, b) \in (Tr^X_{A,B}((id_B \otimes h) \circ f))$$

we then have that $\exists\ x \in X$ satisfying $(a||x, b||x) \in ((id_B \otimes h) \circ f)$, but this implies there's an $x' \in X$ such that

$$(a\|x, b\|x') \in f \text{ and}$$
$$(x', x) \in h.$$

Therefore, we also have that $(a\|x', a\|x) \in (id_A \otimes h)$, and

$$(a\|x', b\|x') \in (f \circ (id_A \otimes h))$$
$$\iff (a, b) \in (Tr^X_{A,B}(f \circ (id_A \otimes h))).$$

This proves naturality in the traced line.

Naturality in the untraced variables is exactly the Tightening coherence condition in 4.4.1 below. This condition ensures that composing with the untraced variables before or after taking the trace gives the same result.

4.4. Coherence Conditions

In [7], Hasegawa defines four coherence conditions that any candidate function must fulfill in order to be a trace.

4.4.1. Tightening

Satisfying this condition ensures the trace is natural with respect to the inputs and outputs not taking part in the trace. This idea is encapsulated in the following equality. Let $f \subseteq (A \otimes X) \times (B \otimes X)$, $h \subseteq A' \times A$, $k \subseteq B \times B'$, and id_X be the identity relation on X.

$$Tr^X_{A',B'}((k \otimes id_X) \circ f \circ (h \otimes id_X)) = k \circ Tr^X_{A,B}(f) \circ h \quad (4)$$

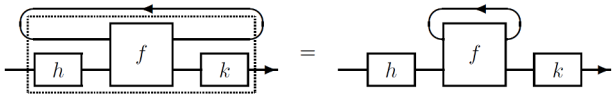


Figure 12: *Tightening. [7]*

Suppose that (a', b') is an element of the LHS of equation (4). By the trace condition we know $\exists\ x \in X$ satisfying

$$(a'\|x, b'\|x) \in ((k \otimes id_X) \circ f \circ (h \otimes id_X)).$$

Due to the fact that we're composing relations we know these three inclusions hold true $(a', a) \in h$, $(b, b') \in k$ and $(a\|x, b\|x) \in f$, for some $a \in A$ and $b \in B$. This last inclusion is exactly the condition we're looking for so $(a', b') \in (k \circ Tr^X_{A,B}(f) \circ h)$. This means we have $LHS \subseteq RHS$ from equation (4).

To get the reverse inclusion $RHS \subseteq LHS$ you can simply reverse the above logic. Furthermore, as the choices of f, h and k were arbitrary we can conclude that the tightening condition holds.

4.4.2. Yanking

The original condition is stated for balanced monoidal categories. Luckily for us, it can be simplified when working with Props because they are symmetric monoidal categories. Namely, the twists are identities, so we need not worry about them here, and for the braids we have $(c_{A,B})^{-1} = c_{B,A}$.

Therefore, we just need to show that the trace of $c_{X,X} : X \otimes X \to X \otimes X$, defined by $(\bar{x_1}\|\bar{x_2}) \mapsto (\bar{x_2}\|\bar{x_1})$, satisfies

$$Tr^X_{X,X}(c_{X,X}) = id_X.$$

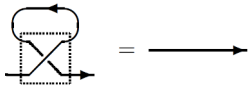


Figure 13: *Yanking. [7]*

To see this suppose $(x_1, x_2) \in (Tr^X_{X,X}(c_{X,X}))$ and assume for contradiction that $x_1 \neq x_2$. The definition of the trace implies $\exists\ x \in X$ such that

$$(x_1\|x, x_2\|x) \in c_{X,X}$$

by the definition of the symmetry we get $x_2 = x = x_1$, a contradiction. That is, $(Tr^X_{X,X}(c_{X,X})) \subseteq id_X$.

For the opposite inclusion, let $(x, x) \in id_X$. Note that $(x\|x, x\|x) \in c_{X,X} \iff (x, x) \in Tr^X_{X,X}(c_{X,X})$. So, $id_X \subseteq (Tr^X_{X,X}(c_{X,X}))$. Together with the above inclusion we have that $Tr^X_{X,X}(c_{X,X}) = id_X$ as required.

4.4.3. Superposing

This condition, realized in the Prop of relations is as follows, let $f \subseteq (A \otimes X) \times (B \otimes X)$ be a signal relation and id_C be the identity on C. Then, the superposing condition states

$$Tr^X_{C\otimes A, C\otimes B}(id_C \otimes f) = id_C \otimes Tr^X_{A,B}(f).$$

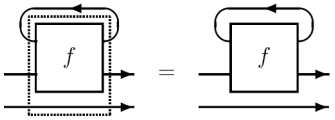


Figure 14: *Superposing. [7]*

To see that this holds, let $a \in A$, $b \in B$, $c_1, c_2 \in C$. If

$$(c_1\|a, c_2\|b) \in (Tr^X_{C\otimes A, C\otimes B}(id_C \otimes f)),$$

then $(c_1\|a\|x, c_2\|b\|x) \in id_C \otimes f$ for some $x \in X$. That is, $(a\|x, b\|x) \in f$ and $c_1 = c_2$. taking the trace of f we get

$$(c_1\|a, c_2\|b) = (c_1\|a, c_1\|b) \in id_C \otimes (Tr^X_{A,B}(f)).$$

As c, a and b were arbitrary this implies.

$$(Tr^X_{C\otimes A, C\otimes B}(id_C \otimes f)) \subseteq (id_C \otimes (Tr^X_{A,B}(f))).$$

For the opposite inclusion, suppose for some $c \in C$ that

$$(c\|a, c\|b) \in id_C \otimes (Tr^X_{A,B}(f)).$$

By the definition of trace, this implies that, for some $x \in X$, $(a\|x, b\|x) \in f$, i.e.
$(c\|a\|x, c\|b\|x) \in id_C \otimes f$. By taking the trace with respect to X, we get

$$(c\|a, c\|b) \in (Tr^X_{C\otimes A, C\otimes B}(id_C \otimes f)).$$

Noting that $(c\|a, c\|b)$ was arbitrary, we get the required inclusion. Together with the previous inclusion we get that $Tr^X_{C\otimes A, C\otimes B}(id_C \otimes f) = id_C \otimes Tr^X_{A,B}(f)$ as required.

4.4.4. Exchange

This condition ensures that order in which you trace your inputs doesn't matter. I will slightly alter this conditions from that found in ([7]) by replacing $c^{-1}_{Y,X}$ with $c_{X,Y}$. Which is a valid move as the definition of symmetric monoidal categories states that the two are equal.

$$Tr^X_{A,B}(Tr^Y_{A\otimes X, B\otimes X}(f)) =$$
$$Tr^Y_{A,B}(Tr^X_{A\otimes Y, B\otimes Y}((id_B \otimes c_{Y,X}) \circ f \circ (id_A \otimes c_{X,Y})))$$

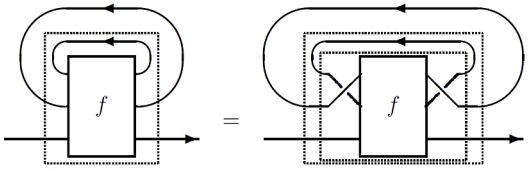


Figure 15: *Exchange. [7]*

where $c_{Y,X} : Y \otimes X \to X \otimes Y$, defined by $(y||x) \mapsto (x||y)$, $\forall (y||x) \in Y \otimes X$, is the symmetry on $Y \otimes X$, id_B and id_A are the identity relations on A and B respectively, and $f \subseteq (A \otimes X \otimes Y) \times (B \otimes X \otimes Y)$.

I claim the above equation holds true for (3). To see this, suppose $\exists\, a \in A, b \in B$ such that

$$(a, b) \in (Tr^X_{A,B}(Tr^Y_{A\otimes X, B\otimes X}(f)))$$

undoing the trace first with respect to X then with respect to Y gives an $x \in X$ and a $y \in Y$ such that

$$(a||x, b||x) \in (Tr^Y_{A\otimes X, B\otimes X}(f)), \text{ and}$$
$$(a||x||y, b||x||y) \in (f).$$

By applying the suitable symmetries to the $X \otimes Y$ inputs and outputs of f, we also have $(a||y||x, b||y||x) \in ((id_B \otimes c_{Y,X}) \circ f \circ (id_A \otimes c_{X,Y}))$. We can now take the traces with respect to X first and Y second to get

$$(a, b) \in Tr^Y_{A,B}(Tr^X_{A\otimes Y, B\otimes Y}((id_B \otimes c_{Y,X}) \circ f \circ (id_A \otimes c_{X,Y}))).$$

This proves that $LHS \in RHS$.

For the opposite inclusion, $RHS \in LHS$ simply reverse the above argument and apply the inverse symmetries

We can now rest assured that $Rel_{\mathbb{S}}$ equipped with (3) is a traceable Prop.

4.5. Trace Example

Now that we have a trace let's apply it to something. As functions are special cases of relations, there will be a trace for any function on the signal tuples. We will use the function $\delta_X : X \to X \otimes X$ defined by $\delta_X(\mathbf{x}) := \mathbf{x}\|\mathbf{x}$. This function is obviously reactive. The trace of δ_X, however, is not a function.

To see this, consider $Tr^X_{0,X}(\delta_X) \subseteq 0 \times X$. Suppose $\mathbf{y} \in X$ then the pair $(e, \mathbf{y})$ is an element of the trace if $(e||\mathbf{x}, \mathbf{y}||\mathbf{x}) \in \delta_X$ for some $\mathbf{x} \in X$, obviously $\mathbf{x} = \mathbf{y}$ satisfies the trace condition. But as $\mathbf{y}$ was arbitrary this means (e, y) is an element of the trace for every element $\mathbf{y} \in X$. Thus, the trace of this function is a one to many relation, and not a function.

However, all is not lost. If we compose the traced line with a one-sample delay as shown $(id_X \otimes \Delta_x) \circ \delta_X$. As with the previous example, when we take the trace $Tr^X_{0,X}((id_X \otimes \Delta_x) \circ \delta_X)$, $(e, \mathbf{y})$ is an element of the relation if there's an $\mathbf{x}$ such that $(e\|\mathbf{x}, \mathbf{y}\|\mathbf{x})$. But this time we know the first value of the traced line is $\mathbf{0}$, because of the delay. This implies the only value that can satisfy the trace condition is $\mathbf{0} \in X$, the zero signal. Thus, the delayed version of δ_X does in fact trace to a function. This example is part of a more general phenomenon. In the next section we will prove that delaying any reactive function is enough to ensure that trace is a reactive function.

5. TRACE OF DELAYED REACTIVE FUNCTIONS

In this section we will see that given any reactive function we need only alter its output slightly with a delay to obtain a reactive function that remains reactive and functional when we take its trace.

Definition: Delayed Reactive Function

Given a reactive function $g : X \otimes Y \to X \otimes Z$, its corresponding delayed reactive function is $f : X \otimes Y \to X \otimes Z$ defined as

$$f := (\Delta_X \otimes id_Z) \circ g \tag{5}$$

that is, f acts the same as g except that it adds a delay of one time step to the top X-tuple of g's output streams. Furthermore, f is reactive by the results of section 3.3 and lemma 1.

I will use $tr(f) : Y \to Z$ to refer to the trace of f, instead of $Tr^X_{Y,Z}(f) : Y \to Z$. I do this to avoid cumbersome notation and because it is obvious which lines are being traced.

Theorem 1. *The trace of a delayed reactive function is a reactive function.*

The following proof is divided into three parts. Parts 1 and 2 show that the $tr(f)$ is a function by displaying the properties of existence and uniqueness respectively. Part 3 shows that it is reactive.

5.1. Proof Part 1: Existence of the Trace

Here we will prove the existence of $tr(f)(y)$ for all y in Y. Remember that for $tr(f)(y)$ to exist there must be an $x^* \in X$ satisfying the trace condition:

$$f(x^*||y) = x^*||z \tag{6}$$

whence $z = tr(f)(y)$. So, the goal here is to show that for each $y \in Y$ there's an $x^* \in X$ satisfying (6).

We fix $y \in Y$ and the proceed to inductively cook up a suitable x^* as follows

$$x^* \upharpoonright n := \begin{cases} 0, & n = 0, \\ 0, g(x^* \upharpoonright (n-1)||y \upharpoonright (n-1))[1:X], & n > 0. \end{cases} \tag{7}$$

That is we define $x^* \upharpoonright 0 := 0$ and inductively calculate $g(x^* \upharpoonright (n-1)||y \upharpoonright (n-1))$ to build $x^* \upharpoonright n$. Before we move on, we must check that the reverse direction holds true that the next inductive step does in fact contain the current one, i.e. $(x^* \upharpoonright n+1) \upharpoonright n = x^* \upharpoonright n$. The following show this to be true

$$\begin{aligned}(x^* \upharpoonright n+1) \upharpoonright n &= (0, g(x^* \upharpoonright n||y \upharpoonright n)[1:X]) \upharpoonright n \\ &= 0, g(x^* \upharpoonright (n-1)||y \upharpoonright (n-1))[1:X] \\ &= x^* \upharpoonright n\end{aligned}$$

The first and last equalities come from (7) and the second holds true because the object being restricted to n values is $n+1$ values long and by g's reactivity.

I claim that x^* satisfies (6). To see this, we first check the trace condition holds for $n = 0$:

$$\begin{aligned}f(x^* \upharpoonright 0||y \upharpoonright 0)[1:X] &= \Delta(g(x^* \upharpoonright 0||y \upharpoonright 0)[1:X]) \\ &= \Delta(g(x^*||y) \upharpoonright 0[1:X]) \\ &= 0, g(x^*||y) \upharpoonright (-1)[1:X] \\ &= 0 \\ &= x^* \upharpoonright 0.\end{aligned}$$

In fact, the above is true for all $x \in X$, the top X tuple in the output is always 0. Next, we check the same for $n > 0$:

$$\begin{aligned}f(x^* \upharpoonright n||y \upharpoonright n)[1:X] &= \Delta(g(x^* \upharpoonright n||y \upharpoonright n)[1:X]) \\ &= \Delta(g(x^*||y) \upharpoonright n[1:X]) \\ &= 0, g(x^*||y) \upharpoonright (n-1)[1:X] \\ &= 0, g(x^* \upharpoonright (n-1)||y \upharpoonright (n-1))[1:X] \\ &= x^* \upharpoonright n.\end{aligned}$$

The above two systems of equalities show that x^* satisfies the trace condition for y. Moreover, as the choice of y was arbitrary, we can conclude that for every element of Y there is an x^* satisfying the trace condition. This implies that $tr(f)(y)$ exists for all y.

5.2. Proof Part 2: Uniqueness of the Trace

In this part, we discover that the x^* defined in Part 1 is unique in its ability to satisfy (6) and conclude that $tr(f)$ is a function.

Suppose, for contradiction, that there is an $x^\dagger \in X$ such that $x^\dagger \neq x^*$ and $f(x^\dagger||y) = x^\dagger||z^\dagger$. Then, we can find a smallest index $n \in \mathbb{N}$ such that $x^\dagger \upharpoonright n \neq x^* \upharpoonright n$.

By minimality of n it must be true that:

$$x^\dagger \upharpoonright (n-1) = x^* \upharpoonright (n-1),$$

but that then implies the following:

$$\begin{aligned}x^\dagger \upharpoonright n &= 0, g(x^\dagger \upharpoonright (n-1)||y \upharpoonright (n-1))[1:X] \\ &= 0, g(x^* \upharpoonright (n-1)||y \upharpoonright (n-1))[1:X] \\ &= x^* \upharpoonright n.\end{aligned}$$

This is obviously a contradiction. One can recycle the above argument to disprove any candidate smallest index of inequality, and so conclude that $x^\dagger \upharpoonright n = x^* \upharpoonright n$ for all $n \in \mathbb{N}$, i.e. that $x^\dagger = x^*$, thus our assumption that x^* wasn't unique was incorrect.

It's now easy to see that setting $tr(f)(y) = z$ from 6 defines a relation that has the existence and uniqueness properties, and as such defines a function $tr(f) : Y \to Z$.

5.3. Proof Part 3: Reactivity of the Trace

In this part we show that $tr(f)$ has the reactive property as well.

Suppose, we have $x_1, x_2 \in X$, $y_1, y_2 \in Y$ and $n \in \mathbb{N}$ such that the following are true

$$\begin{aligned}x_1 \upharpoonright n &= x_2 \upharpoonright n \\ y_1 \upharpoonright n &= y_2 \upharpoonright n,\end{aligned}$$

then, by reactivity of f

$$\begin{aligned}tr(f)(y_1) \upharpoonright n &= f(x^*||y_1) \upharpoonright n[X+1:X+Z] \\ &= f(x^*||y_2) \upharpoonright n[X+1:X+Z] \\ &= tr(f)(y_2) \upharpoonright n.\end{aligned}$$

By combining Parts 1, 2 and 3 we can see that for a reactive function f of the form described in equation (5) taking its trace results in reactive function. That is, we can augment any reactive function we want to trace by putting a delay on the variable that we wish to trace to ensure that we get a reactive function as output. **QED**

To see that the reactivity property is needed, consider the function $\rho_M : M \to M$ defined by $\rho(\mathbf{x})(t) = \mathbf{x}(t+1)\ \forall\ t \in \mathbb{Z}$. Semantically, this function predicts the future of given signal. Importantly, ρ is not reactive, to see this suppose we have $\mathbf{x} = \ldots, \mathbf{0}_{-1}, \mathbf{1}_0, \mathbf{1}_1, \mathbf{1}_2, \mathbf{1}_3, \ldots$ and $\mathbf{y} = \ldots, \mathbf{0}_{-1}, \mathbf{1}_0, \mathbf{1}_1, \mathbf{0}_2, \mathbf{0}_3, \ldots$ in M, obviously $\mathbf{x} \upharpoonright 1 = \mathbf{y} \upharpoonright 1$. However,

$$\begin{aligned}\rho(\mathbf{x}) \upharpoonright 1 &= (\ldots, \mathbf{1}_{-1}, \mathbf{1}_0, \mathbf{1}_1, \mathbf{1}_2, \ldots) \upharpoonright 1 = \mathbf{1}_0, \mathbf{1}_1 \\ &\neq \mathbf{1}_0, \mathbf{0}_1 = (\ldots, \mathbf{1}_{-1}, \mathbf{1}_0, \mathbf{0}_1, \mathbf{0}_2, \ldots) \upharpoonright 1 \\ &= \rho(\mathbf{y}) \upharpoonright 1.\end{aligned}$$

Moreover, $\Delta_M \circ \rho_M = id_M$, as the time-step wise definitions of of the two functions cancel out (cf. section 2.4). Therefore, for δ_M as defined in Section (4.5), we have the following

$$((\Delta_M \circ \rho_M) \otimes id_M) \circ \delta_M = \delta_M$$

But, from section (4.5) we already know that the trace of this function is a not a function. This exemplifies how important reactivity is in having the trace return a reactive function.

6. UNDERSTANDING RECURSION

To summarise the last two sections, in Section 4 we augmented the Prop $Rel_{\mathbb{S}}$ with a trace to model feedback and loops, and in the last section we proved that this trace maps delayed reactive functions to reactive functions. We can now deconstruct the Recursion operator from FAUST and show that it preserves reactivity and functionality.

Recall the definition of Recursion from Section 2. One can identify four necessary signal processors from this definition. It needs two compatible processors, let these be the reactive functions f :

$Y + X \to Z + W$ and $g : W \to X$. It also needs a processor to duplicate the top W outputs of f, let this be the reactive function $\gamma_{2,W} : W \to W + W$. Lastly, it needs a delay to ensure that taking the trace will return a reactive function, let this be the reactive function $\Delta_X : X \to X$. With all this in mind, consider the following function $h : Y + X \to Z + W + X$, a combination of all these signal processors

$$h = (id_{Z+W} \otimes \Delta_X) \circ (id_{Z+W} \otimes g) \circ (id_Z \otimes \gamma_{2,W}) \circ f,$$

This is a reactive function because all of its components are reactive functions which have been stitched together with operators that preserve reactivity and functionality. Moreover, h is a delayed reactive function, and so $Tr^X_{Y,Z+W}(h)$ will be a reactive function too.

One can verify that h satisfies the FAUST definition of Recursion, because g feeds back into f via a one-sample delay, the W output of f is duplicated to supply g and to rejoin f's Z output to form the overall output. This is exactly The action described by Recursive composition in Section 2. So we have

$$f \sim g = Tr^X_{Y,Z+W}((id_{Z+W} \otimes (\Delta_X \circ g)) \circ (id_Z \otimes \gamma_{2,W}) \circ f). \quad (8)$$

And we have that for any compatible reactive functions f, g, i.e. signal processors, $f \sim g$ exists, is unique and is a reactive function itself. So we can conclude that Recursive composition will always return a valid signal when given two valid signal processors. Furthermore, it is obvious now that the definition needs a delay to ensure the trace gives a reactive function.

7. CONCLUSION

We have seen that the Block Diagram Algebra can be extended to a categorical approach, namely we can model it using a traced version of $Rel_{\mathbb{S}}$. Where this category is a Traced Prop of Relations on Signals. Using this approach we have seen that each of the five main FAUST operators will always give a valid signal processor because they preserve two key properties: functionality and reactivity.

We have also witnessed a proof of the theorem that all delayed reactive functions trace to reactive functions. This result showed that definition of the Recursion operator needs a 1-sample delay to ensure that when the trace is taken it will return a reactive function (aka a signal processor).

Furthermore, this categorical interpretation of the algebra provides evidence that the Sequential and Parallel operators are more atomic than Merge and Split, because Merge and Split can be decomposed into primitives stitched together by Sequential Parallel. The categorical interpretation also provides evidence that the Recursion operator is fundamentally different from the other four, because it requires a trace.

More research and investigations need to be conducted to ascertain whether this approach can be applied to more concrete problems such as optimisation.

8. ACKNOWLEDGMENTS

I owe a great debt of gratitude to my dissertation supervisor Guy McCusker whose help and advice was instrumental!

9. REFERENCES

[1] Yann Orlarey, Dominique Fober, and Stéphane Letz, "An Algebra for Block Diagram Languages," in *International Computer Music Conference*, ICMA, Ed., Gothenburg, Sweden, 2002, pp. 542–547.

[2] André Joyal, Ross Street, and Dominic Verity, "Traced monoidal categories," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 119, no. 3, pp. 447–468, 1996.

[3] Filippo Bonchi, Joshua Holland, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi, "Diagrammatic algebra: from linear to concurrent systems," *Proceedings of ACM on programming languages*, vol. 3, no. POPL, pp. 1–28, 2019.

[4] Bob Coecke and Aleks Kissinger, "Picturing quantum processes," in *Diagrammatic Representation and Inference*, Peter Chapman, Gem Stapleton, Amirouche Moktefi, Sarah Perez-Kriz, and Francesco Bellucci, Eds., Cham, 2018, pp. 28–31, Springer International Publishing.

[5] David Sprunger and Bart Jacobs, "The differential calculus of causal functions," *arXiv preprint arXiv:1904.10611*, 2019.

[6] Faust Website Developers, "Faust manual: syntax and documentation," .

[7] MASAHITO HASEGAWA, "On traced monoidal closed categories," *Mathematical structures in computer science*, vol. 19, no. 2, pp. 217–244, 2009.

WHAT'S NEW IN THE FAUST ECOSYSTEM AND COMMUNITY?

Stéphane Letz
Univ Lyon, GRAME-CNCM, INSA Lyon, Inria, CITI, EA3720, 69621 Villeurbanne, France
letz@grame.fr

Romain Michon
Univ Lyon, Inria, INSA Lyon, CITI, EA3720, 69621 Villeurbanne, France
romain.michon@inria.fr

Yann Orlarey
Univ Lyon, GRAME-CNCM, INSA Lyon, Inria, CITI, EA3720, 69621 Villeurbanne, France
orlarey@grame.fr

ABSTRACT

This paper presents an overview all the new developments and contributions in the FAUST programming language since the 2020 International FAUST Conference. It shows a growing and dynamic community with artistic projects, plugins, standalone applications, integration in audio programming environments, development tools, research projects, embedded devices, Web applications, etc., produced by a large variety of contributors.

1. INTRODUCTION

The FAUST [1] community is growing steadily. This paper summarizes all the activity that have taken place since IFC 2020. Section 2 details new developments done in the compiler, architectures files, and various complementary tools. Section 3 explains efforts conducted to improve the documentation. Section 4 lists the various places and programs where FAUST is taught and part of the curriculum. Section 5 details new recently added libraries. Section 6 present the FAST (Fast Audio Signal-processing Technologies on FPGA) project. And finally, section 7 is about the tools that have been implemented to strengthen the FAUST community itself.

2. DEVELOPMENTS

2.1. New Backends

Three new backends have been developed. They allow us to use DSP programs in a larger set of targets and to reach new communities.

2.1.1. DLang + faust2dplug

D (Dlang) [1] is a general-purpose programming language with static typing, systems-level access, and C-like syntax. A backend and several architecture files have been contributed by Ethan Recker, with the help of GRAME.

The goal is to generate D code for Dplug, [2] an open-source audio plug-in framework existing since 2013 and allowing for the production of VT23, VST3, AUv2, AAX, and LV2 plugins for macOS, Windows, and Linux. A `faust2dplug` script has then been developed to simplify the production of Dplug projects starting from the FAUST DSP source code.

[1] `https://dlang.org`
[2] `https://dplug.org`

2.1.2. CSharp

C# is a general-purpose, multi-paradigm programming language used in some games engines. A backend and several architecture files have been contributed by Mike Oliphant, with the help of GRAME.

2.1.3. Julia

Julia [3] [2] is a general-purpose programming language for scientific computing that appeared in 2012. It uses an LLVM-based compiler, a dynamic type system with parameterized polymorphism and distributed parallel execution. It can be easily interfaced with existing languages like C, Fortran, or Python. For the programmer, it comes with a REPL (Real Eval Print Loop) model allowing for a simple and fast interaction with the system.

An integration of the libfaust compiler into Julia was first developed by Cora Johnson-Roberson. [4] A Julia backend was then added to the FAUST compiler. It allows us to generate Julia code on the fly from any FAUST DSP program to then compile it and run it directly in the Julia environment. A set of architecture files (`meta.jl`, `UI.jl`, `MapUI.jl`, `GTKUI.jl`, `OSCUI.j`) were implemented to:

- control the DSP with a graphical interface or an OSC interface
- interface it to an audio layer based on the PortAudio project (with the `audio.jl` and `portaudio.jl` files).

Test tools have also been developed:

- the `minimal.jl` file shows how the generated Julia code can be used in a minimal program that allocates and instantiates the DSP, and calls the compute function. The `MapUI.jl` file is used to eventually control the DSP. The command line [5] should be used to create a `foo.jl` file ready to be tested,
- the `faust2portaudiojulia` tool transforms a FAUST DSP program into a fully functional Julia source file that uses the PortAudio library for real-time audio rendering, and can be controlled with OSC messages. By default, it starts with the GTK-based GUI. It uses the `MapUI.jl`, `OSCUI.jl` and `GTKUI.jl` architecture files.

The Julia ecosystem contains over 4,000 packages that are registered in a general registry.[6] A `Faust.jl` package originally de-

[3] `https://julialang.org`
[4] `https://github.com/corajr/Faust.jl`
[5] `faust -lang julia -a julia/minimal.jl foo.dsp -o foo.jl`
[6] `https://juliahub.com`

veloped by Cora Johnson-Roberson has been extended.[7] It allows for the use of the libfaust library and the FAUST backend of Julia.

This groundwork will facilitate further developments in the area of machine learning applied to audio and DSP, where the Julia environment and language are increasingly used. The integration of libfaust in Julia allowed – for example – Cora Johnson-Roberson to do some experiments using neural networks [8]. We can imagine in the long run more advanced integration in this domain with the FAUST backend directly producing Julia code.

2.2. Modular Synthesis

Modular synthesizers are made of separate modules with different functions. Modules can be connected together using cables or a matrix connection system. The outputs (voltages) of the modules can function as (audio) signals, control voltages, or logic/time conditions. Typical modules are wave generators, effects, envelope generators, etc.

While modules were traditionally implemented using analog electronic circuits, digital modules are slowly becoming the norm. Such systems usually combine a processor (e.g., microcontroller, DSP chip, etc.) and an audio ADC/DAC. Their form factor and interface is standardized (following the Eurorack standard,[9] for example). They are then assembled in racks, and controlled by an external patching system.

Applications like VCV-Rack [10] allow a software emulation of this modular synthesis principle. The plugins are then developed in C++ with an SVG-based interface.

In this context, developments have been carried out to allow for the prototyping of plugins for the VCV-Rack format using the FAUST programming language.

2.2.1. The faust2vcvrack Tool

The `faust2vcvrack` tool compiles a FAUST DSP program into a folder containing (i) the C++ source code of the VCV-Rack plugin and (ii) a MakeFile to compile it. By default, the resulting C++ code is then compiled and installed in the VCV-Rack application.

The FAUST DSP code classically produces audio signals in the range [-1..1]. Since the VCVs expect audio signals in the range [-5v..5v], they are automatically converted in the architecture file. CV commands in the range [0v..10v] are assigned to the [min..max] range of the controllers.

Polyphonic modules can be created using the `-nvoices <n>` parameter up to 16 voices. The freq/gate/gain convention[11] can be used in the DSP code. VCV Rack follows the 1V/octave convention for MIDI pitch values, so MIDI signals are automatically converted to freq using this convention. Gain and gate signals (using the [0v..10v] range) are converted to [0..1] values.

Controllers (typically buttons, sliders, or bar graphs) are automatically transformed into GUI elements (like switches, buttons, or leds). But they can also be connected to CV inputs/outputs by using a `[CV:N]` metadata used in the input (typically sliders or nentry) or output (typically bargraphs) controllers to connect them to the CV signal instead of the GUI parameters.

[7] `https://github.com/sletz/Faust.jl`

[8] `https://github.com/corajr/faust_nn`

[9] `https://en.wikipedia.org/wiki/Eurorack`

[10] `https://vcvrack.com`

[11] `https://faustdoc.grame.fr/manual/midi/#midi-polyphony-support`

2.2.2. VCV Prototype

The VCV Prototype Module runs scripting languages for prototyping, learning, and live coding. It can currently be programmed using JavaScript, Lua, Vult, or PureData. A generic GUI with 6 inputs/outputs (either audio or CV signals), 6 knobs, 6 lights (RGB LEDs) or 6 switches (with RGB LEDs) is defined.

FAUST support has been added thanks to libfaust embedding the Interpreter backend. It allows us to edit/compile/execute DSP programs on the fly, with acceptable performances (even if using the LLVM JIT would allow us to generate faster code, but at the expense of a much more complicated installation procedure).

2.3. Embedded Platforms

2.3.1. The Daisy Board

Daisy[12] is an embedded platform for music developed by Electrosmith. It combines on a single board a low-consumption ARM embedded processor, SDRAM memory, as well as a stereo audio codec. Several boards hosting the Daisy Seed[13] have been developed on top of the same chip.

They are distributed with a software library which abstracts low-level embedded software development for the boards and provide support for a number of languages including C++, Arduino, and Max/MSP Gen~. Specific developments have been carried out to program this new device with FAUST.

2.3.2. The faust2daisy Tool

The `faust2daisy` tool compiles a Faust DSP program into a folder containing the C++ source code and a Makefile to compile it. Options to compile polyphonic DSP and add MIDI control are available. Specific architecture files have been written:

- `faust/gui/DaisyControlUI.h`: to be used with the DSP `buildUserInterface` method to implement button, checkbox, hslider, vslider controllers, and interpret the specific metadata used to describe the hardware,
- `faust/midi/daisymidi.h`: implements a `midi_handler` subclass to decode incoming MIDI events.

The current `faust2daisy` tool can only be used to program the POD[14]. On this board, the 2 switches and 2 knobs can be connected to UI controllers using metadata.

The programming model still needs to be extended to support more devices and better support their heterogenous memory model.

2.4. Exported Box and Signal API

The FAUST compiler can be used in applications or plugins using the libfaust library, embedding the LLVM backend and allowing DSP code to be dynamically compiled and executed. A set of C and C++ headers are available to access the API.

Work has been done to give access to other intermediate points in the compilation chain, so that new use-cases can be considered, like developing graphical language interfaces or connecting with machine learning models.

[12] `https://www.electro-smith.com/daisy`

[13] `https://www.electro-smith.com/daisy/daisy`

[14] `https://www.electro-smith.com/daisy/pod`

The compilation chain of the FAUST compiler is composed of several steps (see Figure 1):

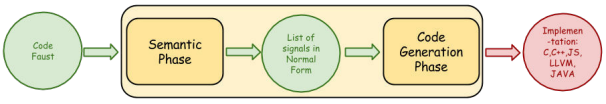


Figure 1: *The compilation chain.*

Starting from the DSP source code, the Semantic Phase produces signals as conceptually infinite streams of samples or control values. Those signals are then compiled in imperative code (C/C++, LLVM IR, WebAssembly, etc.) in the Code Generation Phase.

The Semantic Phase itself is composed of several steps (see Figure 2):

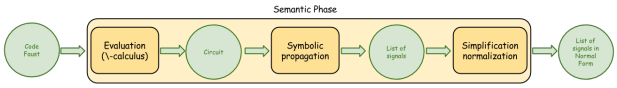


Figure 2: *The semantic phase.*

The initial DSP code using the Block Diagram Algebra (BDA) is translated in a flat circuit in normal form in the Evaluation, lambda-calculus step. The list of output signals is produced by the Symbolic Propagation step. Each output signal is then simplified and a set of optimizations are done (normal form computation and simplification, delay line sharing, typing, etc.) to finally produce a list of output signals in normal form.

The Code Generation Phase translates the signals in an intermediate representation named FIR (FAUST Imperative Representation) which is then converted to the final target language (C/C++, LLVM IR, WebAssembly, etc.) with a set of backends.

2.4.1. Accessing the Box Stage

A new intermediate public entry point has been created in the Semantic Phase, after the Evaluation, lambda-calculus step to allow the creation of a box expression, then beneficiate of all remaining parts of the compilation chain.

It means that a box expression can be *programmatically built* (using the box C++ API, or the C box API version), evaluated to the list of output signals, translated to the FIR format, and finally converted to the final target language, as a ready-to-use C++ class, LLVM or Interpreter factories, to be used with all existing architecture files.

2.4.2. Compiling Box Expressions

To use the box API, the following steps must be taken:

- creating a global compilation context using the `createLibContext` function,
- creating a box expression using the box API, progressively building more complex expressions by combining simpler ones,
- compiling the box expression using the `createCPPDSPFactoryFromBoxes` function to create a DSP factory (or some variants),
- finally destroying the compilation context using the `destroyLibContext` function.

DSP factories allow for the creation of DSP instances, to be used with audio and UI architecture files, outside of the compilation process itself. The DSP instances and factory have to be eventually deallocated when not used anymore.

Here is an example, creating and compiling the DSP code:

```
process = + ~ _;
```

With the box API code:

```
// Create global context
createLibContext();

// Create the box expression
Box box = boxRec(boxAdd(), boxWire());

// Compile it as a factory
string error_msg;
llvm_dsp_factory* factory =
    createDSPFactoryFromBoxes("FaustDSP",
    box, 0, nullptr, "", error_msg);

// Create a DSP instance
dsp* dsp = factory->createDSPInstance();

// Use dsp with all existing architecture
    files
...
// Delete dsp and factory
delete dsp;
deleteDSPFactory(factory);

// Destroy the global context
destroyLibContext();
```

2.4.3. Accessing the Signal Stage

A new intermediate public entry point has been created in the Semantic Phase allowing for the creation of a signal graph (as a list of output signals) then benefiting from all remaining parts of the compilation chain.

The signal C++ API (or the C signal API version) allows us to *programmatically build* the signal graph, then compile it to create a ready-to-use DSP as a C++ class (or LLVM Interpreter factories) to be used with all existing architecture files.

2.4.4. Compiling Signal Expressions

To use the signal API, the following steps must be taken:

- creating a global compilation context using the `createLibContext` function,
- creating signals outputs using the signal API, progressively building more complex expressions by combining simpler ones,

- compiling the list of outputs using the `createCPPDSPFactoryFromSignals` function to create a DSP factory (or some variants),
- finally destroying the compilation context using the `destroyLibContext` function.

The DSP factories allow for the creation of DSP instances, to be used with audio and UI architecture files, outside of the compilation process itself. The DSP instances and factory will finally have to be deallocated once used no more.

Here is an example, using the previous recursive DSP code and the signal API:

```
// Create global context
createLibContext();

// Create the signal expression
tvec signals;
Signal in1 = sigInput(0);
signals.push_back(sigRecursion(sigAdd(
    sigSelf(), in1)));

// Compile it as a factory
string error_msg;
llvm_dsp_factory* factory =
    createDSPFactoryFromSignals("FaustDSP",
    signals, 0, nullptr, "", error_msg);

// Create a DSP instance
dsp* dsp = factory->createDSPInstance();

// Use dsp with all existing architecture
    files
...
// Delete dsp and factory
delete dsp;
deleteDSPFactory(factory);

// Destroy the global context
destroyLibContext();
```

2.4.5. Creating a Language Based on Those APIs

Generating complex expressions by directly using the box or signal APIs can quickly become tricky and impracticable. So a language created on top of them is usually needed. This is exactly what the Block Diagram Algebra is all about, and the entire FAUST language itself.

But giving access to the box and signal APIs allows for completely *new audio languages* to be created, while taking advantage of the compiler infrastructure and existing architectures.

The Elementary audio language, [15] for instance, is built over signal language resembling the one previously described, and uses JavaScript as the upper layer language to help create complex signal graphs programmatically. A similar approach could be proposed with the FAUST signal API. Other approaches using graphical based tools could certainly be tested.

[15] `https://www.elementary.audio`

2.5. Debugging and Optimisation Tools

When FAUST DSP programs are used in demanding projects, the programmer may have to carefully check the behaviour of the generated code, and possibly optimize it as much as possible. Several tools have been developed to help with this.

2.5.1. Debugging the Code

The FIR (FAUST Imperative Representation) backend can possibly be used to look at a textual version of the intermediate imperative language. It displays various statistics, like the number of operations done in the generated `compute` method, or the DSP memory layout.

2.5.2. Using -ct and -cat Options

Using the `-ct` and `-cat` compilation options allows us to check table index range, by verifying that the actual signal range is compatible with the actual table size. Note that since the interval calculation is imperfect, you may see false positives especially when using recursive signals where the interval calculation system will typically produce [-inf, inf] range, which is not precise enough to correctly describe the real signal range.

2.5.3. Using -me Option

Starting with FAUST version 2.37.0, mathematical functions which have a finite domain (like `sqrt` defined for positive or null values, or `asin` defined for values in the [-1..1] range) are checked at compile time when they actually compute values at that time, and raise an error if the program tries to compute an out-of-domain value.

If those functions appear in the generated code, their domain of use can also be checked (using the interval computation system) and the `-me` option will display warnings if the domain of use is incorrect. Note that again, because of the imperfect interval computation system, false positives may appear and should be checked.

2.5.4. Optimizing the Code

Developments have been done to ease the deployment of C++ (or LLVM IR) generated code in real demanding environments.

2.5.5. Compiling for Multiple CPUs

On modern CPUs, compiling native code dedicated to the target processor is critical to obtain the best possible performances. When using the C++ backend, the same C++ file can be compiled with gcc of clang for each possible target CPU using the appropriate `-march=cpu` option.

When using the LLVM backend, the same LLVM IR code can be compiled into CPU specific machine code using the `dynamic-faust` tool. This step will typically be done using the best compilation options automatically found with the `faustbench` tool or `faustbench-llvm` tools. A specialized tool has been developed to combine all the possible options.

2.5.6. The faust2object Tool

The `faust2object` tool [16] either uses the standard C++ compiler or the LLVM dynamic compilation chain (the `dynamic-faust` tool) to compile a FAUST DSP to object code files (.o) and wrapper C++ header files for different CPUs.

The DSP name is used in the generated C++ and object code files, thus allowing to generate distinct versions of the code that can finally be linked together in a single binary.

3. TECHNICAL DOCUMENTATION

3.1. Documentation of the Architecture Files

The FAUST compiler produces the DSP processing code in the form of a module (e.g., a C++ class) that must then be connected to the outside world. The program will be integrated into the audio architecture of the target machine (i.e., a computer, a smartphone, or a web page) and used with a control interface – typically a graphical interface (possibly deported to another machine) – or in the form of gesture controls, if a smartphone with accelerometers or a gyroscope is used for example.

This connection to the outside world is made through what is called an architecture file. Initially developed for our own needs, the architecture files will have to be used by external developers who want to use FAUST in their projects.

An exhaustive documentation about the architecture files and their use has been written to facilitate their work.[17] It covers the classical requirements for deployment on computers, smartphones and tablets, web pages, as well as embedded hardware. The deployment of generators or monophonic effects, but also polyphonic instruments controllable by MIDI is described. The use of existing architecture files is explained, as well as the development of new custom files for specific needs.

3.2. Additional Resources

Additional pages/resources have been added progressively to describe:

- how to integrate the dynamic compiler (in the form of the libfaust library) into programs,[18]
- how to deploy Faust programs on the Web,[19]
- a Frequently Asked Questions (FAQ) page.[20]

4. LEARNING FAUST

FAUST is now taught at several places/institutions in the world.

4.1. Center for Computer Research in Music and Acoustics (CCRMA)

Several courses or tutorials around FAUST are given at CCRMA [21]:

- Julius Smith's FAUST tutorial,[22]
- Romain Michon's FAUST tutorials,[23]
- Music 250a (Physical Interaction Design for Music) course which hosts various tutorials on FAUST and hardware,[24]
- Music 320c (Audio Plugin Development in FAUST and C++),[25]
- Embedded DSP With FAUST Workshop. [26]

4.2. TU Berlin

The regular sound synthesis class at the Audio Communication Group, TU Berlin, makes use of FAUST for exploring the basics of different synthesis algorithms. Student projects based on FAUST include Eurorack modules, standalone drum machines and synthesizers, as well as data sonification approaches. The class is taught by Henrik von Coler, who is director of the Electronic Studio at the TU. HPI Potsdam

The class Data Sonification & Opportunities of Sound at Hasso Plattner Institute, University of Potsdam Potsdam, is an interdisciplinary format, exploring the use of sonification and sound synthesis in the context of design thinking, neurosience, and medical applications. The signal processing part is taught by Henrik von Coler.

4.3. Université Paris 8

A 24 hours introduction to FAUST is given by Alain Bonardi during the first semester to undergraduate students (L3, 3rd year after the french 'baccalauréat') as part of the "Programming Languages in Computer Music 1" course offered in the "Music creation with computers" minor.

4.4. Universidad Nacional de Quilmes

Faust / DSP courses in Spanish, prepared by Juan Ramos. They include the classes of the "Update Seminar on Sound, Science and Technology II," held at the National University of Quilmes (Argentina) with an intro video [27] and several classes. [28]

4.5. RIM & RAN Professional Masters Program

The RIM & RAN professional Masters programs aim at shaping young professionals in the fields of electronic and digital technologies applied to the arts in the prospect of becoming "Producer in Computer Music" (RIM - Réalisateur en Informatique Musicale) and in Digital Arts (RAN - Réalisateur en Arts Numériques).

These producers play an important role in musical and artistic productions, and work at the interface between software developers, applied computer scientists, composers, artists, etc. and all people likely to integrate video, image, and sound in their activities.

[16] `https://github.com/grame-cncm/faust/tree/master-dev/tools/benchmark#faust2object`
[17] `https://faustdoc.grame.fr/manual/architectures/`
[18] `https://faustdoc.grame.fr/manual/embedding/`
[19] `https://faustdoc.grame.fr/manual/deploying/`
[20] `https://faustdoc.grame.fr/manual/faq/`
[21] `https://ccrma.stanford.edu`
[22] `https://ccrma.stanford.edu/~jos/aspf/`
[23] `https://ccrma.stanford.edu/~rmichon/faustTutorials/`
[24] `https://ccrma.stanford.edu/courses/250a-winter-2022/`
[25] `https://ccrma.stanford.edu/courses/320c/`
[26] `https://ccrma.stanford.edu/workshops/faust-embedded-19/`
[27] `https://www.youtube.com/watch?v=DnBI7r273BE`
[28] `https://www.youtube.com/channel/UCD6aeS3GdkEmt86KUehr8LQ/videos`

Most of the courses about signal processing are given around through the FAUST language (M1 Romain Michon 12h / M2 Yann Orlarey 20h).

4.6. Aalborg University in Copenhagen

FAUST is taught by Romain Michon at Aalborg University in Copenhagen (Denmark) as part two one week workshops opened to Masters students of the Sound and Music Computing (SMC) masters program of the medialogy department. The first workshop typically happens in the Fall and focuses on the use of FAUST for programming embedded systems for real-time audio applications. The second workshop takes place in the Spring and is about physical modeling of musical instruments in FAUST.

5. LIBRARIES

5.1. Standard Libraries

Among the significant contributions received in the two last years on the Faust Libraries project, [29] we can mention:

- the `fds.lib` library developed by Riccardo Russo allowing the modelling of physical models by finite difference,
- the `wdmodels.lib` library developed by Dirk Roosenburg allowing WDF (Wave Digital Filter) modelling,
- the `aanl.lib` developed by Dario Sanfilippo. This library provides aliasing-suppressed nonlinearities through first-order and second-order approximations of continuous-time signals, functions, and convolution based on antiderivatives. This technique is particularly effective if combined with low-factor oversampling, for example, operating at 96 kHz or 192 kHz sample-rate,
- the `webaudio.lib` contributed by GRAME, implementing WebAudio filters using their C++ version as a starting point, taken from Mozilla Firefox implementation. This work was done to simplify porting audio effects written using the Web Audio API.

5.2. Community Contributions

Several contribution from the community have appeared in the last two years.

5.2.1. abclib library

The abclib library [30] is released by the CICM / MUSIDANSE (Centre de Recherches Informatique et Création Musicale, Paris 8 University) and is the result of 20 years of research, teaching, and creation in mixed music, expressed as a set of codes in the FAUST language.

The main topics addressed are: spatial sound processing and synthesis using ambisonics, multi-channel sound processing, utility objects for mixed music.

[29] `https://faustlibraries.grame.fr`
[30] `https://github.com/alainbonardi/abclib`

5.2.2. Edge of Chaos

This repository [31] contains libraries including some essential building blocks for the implementation of musical complex adaptive systems in FAUST programming.

It includes a set of time-domain algorithms, some of which are original, for the processing of low-level and high-level information as well as the processing of sound using standard and non-conventional techniques.

It also includes functions for the implementation of networks with different topologies, linear, and nonlinear mapping strategies to render positive and negative feedback relationships, and different kinds of energy-preserving techniques for the stability of self-oscillating systems.

5.2.3. realfaust

The realfaust library [32] contains a set of functions representing domain-limited versions of all FAUST primitives and math functions that can potentially generate INF or NaN values.

The goal of the library is to be able to implement DSP networks that, structurally, are free from INF and NaN values. Hence, the resulting programs should be rock-solid during real-time performance and virtually immune to crashes regardless of how mercilessly a network is modulated or how unstable a recursive system is made.

5.2.4. bitDSP-faust

BitDSP [33] is a set of FAUST library functions aimed to help explore and research artistic possibilities of bit-based algorithms.

BitDSP currently includes implementations of bit-based functions ranging from simple bit operations over classic delta-sigma modulations to more experimental approaches like cellular automata, recursive Boolean networks, and linear feedback shift registers.

A detailed overview of the functionality is in the paper "Creative use of bit-stream DSP in FAUST" presented at IFC 2020.

5.2.5. SEAM library

Sustained Electro-Acoustic Music is a project inspired by Alvise Vidolin and Nicola Bernardini. The SEAM libraries [34] have been developed for this project.

6. FAST: A FUNDED RESEARCH PROJECT AROUND FAUST

FAST[35] (Fast Audio Signal-processing Technologies on FPGA) is a research project funded by the Agence Nationale de la Recherche (ANR – the French National Research Agency). It gathers the strengths of GRAME-CNCM,[36] CITI Lab (INSA Lyon),[37] and LMFA (École Centrale Lyon)[38] towards two goals:

[31] `https://github.com/dariosanfilippo/edgeofchaos`
[32] `https://github.com/dariosanfilippo/realfaust`
[33] `https://github.com/rottingsounds/bitDSP-faust`
[34] `https://github.com/s-e-a-m/faust-libraries`
[35] `https://fast.grame.fr/`
[36] `https://www.grame.fr`
[37] `http://www.citi-lab.fr/`
[38] `http://lmfa.ec-lyon.fr/?lang=en`

- facilitate the design of ultra-low latency embedded systems for real-time audio signal processing,
- use such systems in the context of active control of acoustics.

FAUST plays a central role in this project by facilitating the programming of FPGAs for real-time audio signal processing applications. A "FAUST to FPGA" toolchain is in the process of being implemented. It already allows us[39] to run simple FAUST programs on Xilinx-based FPGA boards such as the Digilent Zybo Z7 (see Figure 3) and the Genesys 2, reaching unparalleled audio latency performances [3].

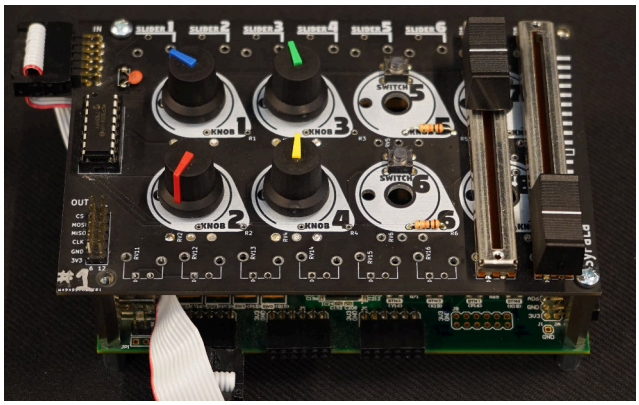


Figure 3: *Digilent Zybo Z7 board equipped with a custom-made control interface designed as part of FAST.*

7. THE FAUST COMMUNITY

7.1. Communication Channels

In addition to existing solutions such as the website, discussion lists and the compiler and applications GitHub development site, more modern communication tools have been put in place:

- a dedicated channel has been created on the Slack collaborative communication platform, organised in channels corresponding to many discussion topics. It currently gathers more than 300 developers (see Figure 4),
- the Audio Programmer website [40] animated by Joshua Hodge hosts a very large community of audio DSP developers. A FAUST channel has been created on this platform, which allows in particular to extend the visibility of the language and its ecosystem beyond the already interested or identified programmers (see Figure 5).

7.2. The "Powered by Faust" Page

A page listing all the significant "Powered with FAUST" projects is maintained: musical pieces or artistic projects, plugins, standalone applications, integration in audio programming environments, development tools, research projects, embedded devices, Web applications, etc are listed.

This page is regularly enriched and as of march 2022, more than 110 projects are described (see Figure 6).

[39] https://github.com/inria-emeraude/syfala

[40] https://theaudioprogrammer.com

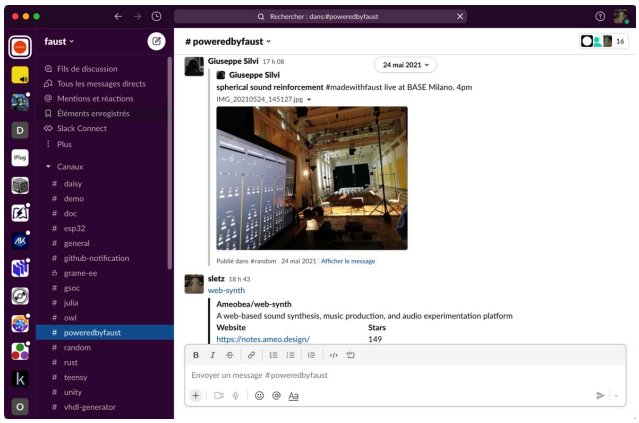


Figure 4: *Faust Slack channel.*

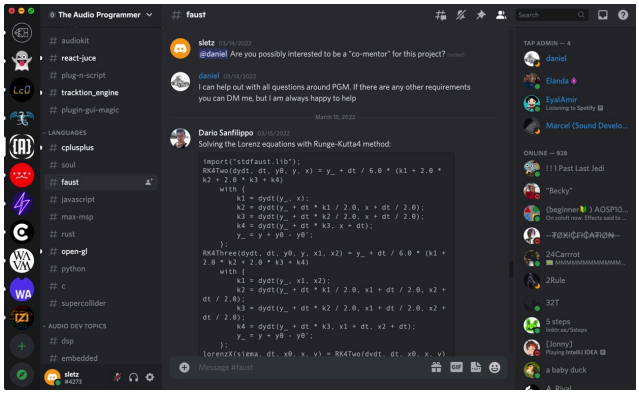


Figure 5: *Faust Discord channel.*

8. ACKNOWLEDGMENTS AND CONCLUSIONS

This paper reflects the richness and diversity of the contributions done in the last two years. Thanks to all contributors for all the different components and projects that have been described!

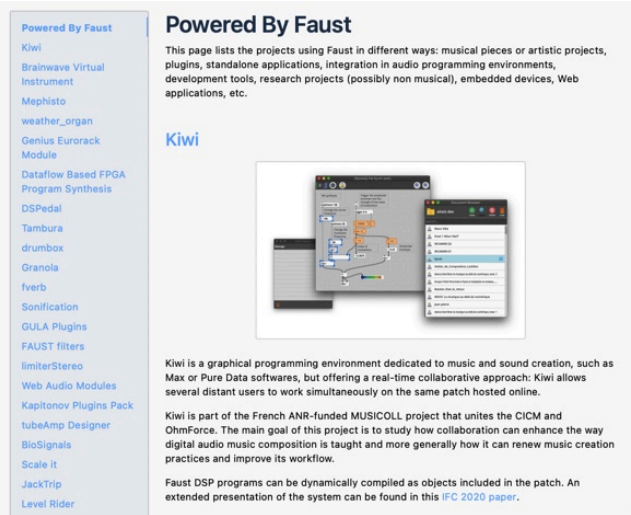


Figure 6: *Part of the "Powered by Faust" list*

9. REFERENCES

[1] Yann Orlarey, Dominique Fober, and Stéphane Letz, "Faust : an Efficient Functional Approach to DSP Programming," in *New Computational Paradigms for Computer Music*, Editions Delatour France, Ed., pp. 65–96. 2009.

[2] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman, "Julia: A fast dynamic language for technical computing," *ArXiv Preprint*, 2012.

[3] Maxime Popoff, Romain Michon, Tanguy Risset, Yann Orlarey, and Stéphane Letz, "Towards an fpga-based compilation flow for ultra-low latency audio signal processing," in *Proceedings of the 2022 Sound and Music Computing Conference (SMC-22)*, Saint-Étienne, France, 2022, *Paper accepted to the conference but not published yet.*

9 782958 412609

19th Sound and Music Computing Conference

Proceedings