The Flexible Audio Source Separation Toolbox Version 2.0

Yann Salaün, Emmanuel Vincent, Nancy Bertin, Nathan Souviraà-Labastie, Xabier Jaureguiberry, Dung T. Tran, Frédéric Bimbot

To cite this version:
Yann Salaün, Emmanuel Vincent, Nancy Bertin, Nathan Souviraà-Labastie, Xabier Jaureguiberry, et al.. The Flexible Audio Source Separation Toolbox Version 2.0. ICASSP, May 2014, Florence, Italy. 2014. hal-00957412

HAL Id: hal-00957412
https://inria.hal.science/hal-00957412
Submitted on 10 Mar 2014

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.
THE FLEXIBLE AUDIO SOURCE SEPARATION TOOLBOX VERSION 2.0

Yann Salaün¹, Emmanuel Vincent¹, Nancy Bertin², Nathan Souvirà-Labastie³, Xabier Jaureguiberry⁴, Dung T. Tran¹, and Frédéric Bimbot²

¹Inria, Villers-lès-Nancy, F-54600, France
²CNRS, IRISA - UMR 6074, Rennes, F-35042, France
³Université Rennes 1, IRISA - UMR 6074, Rennes, F-35042, France
⁴Institut Mines-Télécom, Télécom ParisTech, CNRS LTCI, Paris, F-75014, France
yann.salaun@inria.fr

ABSTRACT

The Flexible Audio Source Separation Toolbox (FASST) is a toolbox for audio source separation that relies on a general modeling and estimation framework that is applicable to a wide range of scenarios. We introduce the new version of the toolbox written in C++, which provides a number of advantages compared to the first Matlab version: portability, faster computation, simplified user interface, more scripting languages. In addition, we provide a state-of-the-art example of use for the separation of speech and domestic noise. The demonstration will give attendees the opportunity to explore the settings and to experience their effect on the separation performance.

1. INTRODUCTION AND MOTIVATIONS

Source separation is one of the major topics in audio signal processing. Recent years have seen a move from blind to guided approaches incorporating more and more knowledge about the sources and/or the mixing process [1].

While most source separation methods are designed for a specific scenario, the flexible audio source separation framework in [2] introduced a compositional approach [3] where the mixture signal is modeled by composing multiple source models together. Each model is parameterized by a number of variables, which may be constrained by the user, trained from separate data or adapted to the considered mixture according to the available information. This framework has been applied to a wide variety of speech and music separation scenarios by exploiting information such as note spectra, cover music recordings, reference speech pronounced by another speaker, or target spatial direction [4–8]. It has also been used as a preprocessing step for instrument recognition, beat tracking, and automatic speech recognition [8,9].

This framework was first implemented in Matlab as version 1.0 of the Flexible Audio Source Separation Toolbox (FASST) which was not demonstrated to the public. Although it facilitates quick prototyping of software modifications, this choice implied large computation time and limited diffusion to communities which do not routinely use Matlab. We introduce the version 2.0 of FASST in C++ which fixes these limitations and provides a simpler user interface. In addition, we provide a new example of use for speech denoising in the context of the 2nd CHiME Challenge [10].

The structure of the rest of the paper is as follows. Section 2 summarizes the framework behind FASST. Section 3 introduces the new implementation and user interface. Future developments are discussed in Section 4.

2. SCIENTIFIC AND TECHNICAL DESCRIPTION

FASST operates in the time-frequency domain. In each time-frequency bin \((n,f)\), the vector \(x_{fn}\) of mixture STFT coefficients recorded at all microphones satisfies

\[
x_{fn} = \sum_{j=1}^{J} y_{jfn}
\]

where \(y_{jfn}\) is the spatial image of the \(j\)th source. The sources are assumed to be zero-mean Gaussian distributed as

\[
y_{jfn} \sim \mathcal{N}(0, v_{jfn} R_{jfn})
\]

where \(v_{jfn}\) denotes the short-term power spectrum of the \(j\)th source and \(R_{jfn}\) its spatial covariance matrix.

The power spectrogram \(V_j\) of each source (i.e., the matrix whose \((f,n)\)th entry is \(v_{jfn}\)) is further factored according to an excitation-filter model followed by two-level nonnegative matrix factorization (NMF). Overall,

\[
V_j = (W_j^{cx} U_j^{ex} G_j^{ex} H_j^{ex}) \odot (W_j^{ft} U_j^{bf} G_j^{ft} H_j^{ft})
\]

where the nonnegative matrices \(W_j^{cx}\), \(U_j^{ex}\), \(G_j^{ex}\) and \(H_j^{ex}\) encode the fine spectral structure, the spectral envelope, the temporal envelope and the temporal fine structure of the excitation, respectively. \(W_j^{ft}\), \(U_j^{bf}\), \(G_j^{ft}\) and \(H_j^{ft}\) encode the same quantities for the resonance filter, and \(\odot\) denotes entry-wise
matrix multiplication. The spatial covariance matrix is itself factored as
\[ R_{ijf} = A_{ijf} A_{ijf}^H \]  
(4)
where the dimension of \( A_{ijf} \) governs the rank of \( R_{ijf} \). Each of the 9 parameters in (3) and (4) may be either fixed or adaptive depending on the information available about, e.g., the spatial position of the sources and the harmonicity of their spectra.

Adaptive parameters are estimated in the maximum likelihood (ML) sense by iteratively fitting the empirical mixture covariance matrix \( \hat{R}_{x,fn} \) using an expectation-maximization (EM) algorithm. The source spatial image signals are eventually obtained by multichannel Wiener filtering. For more details about the algorithm and example settings, see [2].

3. IMPLEMENTATION AND USE

FASST 2.0 is composed of two parts: binary executables written in C++ and user scripts written in Matlab and Python. Code and sound examples are distributed online under the QPL and Creative Commons open source licences\(^1\).

3.1. C++ core

The core of FASST is composed of three C++ programs illustrated in Fig. 1:

1. **Representation**
   The first program takes as input a time-domain mixture WAV file and it computes its time-frequency domain empirical covariance matrix \( \hat{R}_{x,fn} \). Since this matrix is Hermitian, we developed a binary format to save disk space by storing non-redundant entries only.

2. **Parameter estimation**
   The second program performs ML parameter estimation given the above time-frequency domain empirical covariance matrix \( \hat{R}_{x,fn} \). Since this matrix is Hermitian, we developed a binary format to save disk space by storing non-redundant entries only.

3. **Filtering**
   The third program performs Wiener filtering given the input mixture WAV file and the estimated parameters XML file.

These programs rely on standard third-party libraries: libsndfile, Qt, and Eigen. In addition to the portability of C++, FASST 2.0 offers two advantages. Firstly, the parameter estimation code takes full advantage of multicore hardware using OpenMP: computation time can reach a 3x speedup compared to the original implementation in Matlab. Secondly, the user interface has been simplified. Specification of each of the 9 parameters of each source is not required anymore: only those parameters which are adaptive or different from the identity matrix must be specified (e.g., only the basis spectra \( U^x_j \) and the activations \( G^x_j \) in the case of conventional, one-level NMF) and the others are assumed to be fixed and equal to the identity matrix by default.

The demonstration will give attendees the opportunity to manipulate these parameters and to listen to the resulting separation performance.

3.2. User scripts

Similarly to speech recognition software, scripts must be written in order to glue the three core executables together as appropriate for a given source separation problem. The simplest workflow is to initialize the source model, to write it to an XML file, and to call the executables in the above order. More complicated workflows involve multiples call to the executables in order to learn models from separate data before applying them to the considered mixture. FASST 2.0 includes scripts allowing to import/export binary and XML files to/from Matlab and Python. This also increases the portability of the software compared to the original version which included Matlab scripts only.

3.3. Baseline for CHiME

As an advanced example, we provide new scripts for the separation of speech and real-world domestic noise as evaluated in Track 1 of the 2nd CHiME Challenge [10]. The workflow is depicted in Fig. 2. Speech models are trained on separate speaker-dependent data and kept fixed. Noise models are initialized by training from the background noise surrounding the considered mixture and reestimated from the mixture.

Research has been carried regarding model initialization to favor convergence of EM to a relevant local optimum. The basis spectra are initialized by split vector quantization of the

---

\(^1\)http://bass-db.gforge.inria.fr/fasst/
input short-term spectra, the spatial parameters are initialized by splitting the space of directions-of-arrival into regions of equal surface, and the activations on the considered mixture are initialized by the mean activations on the training data. On average, the overall separation quality measured by the signal-to-distortion ratio in decibels (dB) outperforms the best quality reported so far in [11], as shown in Table 1.

4. CONCLUSION AND FUTURE DEVELOPMENT

Thanks to its general modeling and estimation framework, FASST is to our knowledge the most widely applicable source separation software currently available, as exemplified by its use for different scenarios in [4–9]. The demonstration will introduce the advantages of FASST 2.0 compared to the original version in Matlab, which had not been demonstrated to the public so far, and it will allow the audience to interact with the settings. Future planned developments include extending the code for real-time operation and integrating it with automatic speech recognition software using uncertainty propagation.

5. ACKNOWLEDGMENT

This work was supported by the Inria ADT FASST project. We would like to thank O. Rochel, J. Espiau de Lamaëstre, D. Jouvet and G. Gravier for their advice.

6. REFERENCES