Computer Drawing Tools for Assisting Learners, Hobbyists, and Professionals

Emmanuel Iarussi

▶ **To cite this version:**

Emmanuel Iarussi. Computer Drawing Tools for Assisting Learners, Hobbyists, and Professionals. Graphics [cs.GR]. Université Nice Sophia Antipolis, 2015. English. NNT : . tel-01247358v1

HAL Id: tel-01247358

https://inria.hal.science/tel-01247358v1

Submitted on 21 Dec 2015 (v1), last revised 12 Jan 2016 (v2)

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.

UNIVERSITE NICE SOPHIA ANTIPOLIS

DOCTORAL SCHOOL STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

PhD of Science

of the Université Nice Sophia Antipolis

Specialty : COMPUTER SCIENCE

Defended by

Emmanuel IARUSSI

Computer Drawing Tools for Assisting Learners, Hobbyists, and Professionals

Thesis Advisors: George DRETTAKIS & Adrien BOUSSEAU

prepared at INRIA Sophia Antipolis, GRAPHDECO Team

defended on September 29, 2015

Jury :

Reviewers :	Marc ALEXA	-	Professor, Technische Universität Berlin
	Niloy J. MITRA	-	Professor, University College London
Advisor :	George DRETTAKIS	-	Research Director, INRIA (GraphDeco)
CoAdvisor :	Adrien BOUSSEAU	-	Researcher , INRIA (GraphDeco)
President :	Marc ANTONINI	-	Research Director, I3S
Examiner :	Tobias ISENBERG	-	Research Director, INRIA (Aviz)

"Droit devant soi on ne peut pas aller bien loin."
Le Petit Prince, Antoine de Saint-Exupéry,
NY, 1943

Acknowledgments

A large number of people have made this Thesis possible and it is my pleasure to thank them for contributing during these three years trip called PhD.

First of all, I would like to thank *Adrien Bousseau* for his personal support and academic guidance. He has always been available and open for discussion. Adrien has been attentive to every detail and kept me positive even during hard times. From the first day at INRIA I felt he believed I could reach this point and this felling ensured me on every step. I have experienced an amazing growth on his side. I would also like to show my gratitude to *George Dettakis* for letting me join his group and for the pertinent advices I have received from him.

This Thesis greatly benefited from collaborators in different research areas. I would like to thank *Theophanis Tsandilas* for working with me on the Drawing Assistant and guiding me into the HCI world. Also thanks to *David Bommes*, for the countless hours in front of the blackboard. I owe him all the beautiful geometry knowledge I have now. Also, it has been a pleasure for me to be an intern at Adobe Research under *Wilmot Li* supervision. I'll always have great memories of the time spent in San Francisco. Thanks Wil for the opportunity, but mostly for being a great guy to work with.

I would also like to thank *Marc Alexa* and *Niloy Mitra* for accepting being reviewers for this Thesis. I've felt inspired by their work along these years and it's a great honor to have them in my Jury. I am also grateful to all my colleagues I have had in both REVES and GRAPHDECO teams. Thanks for the ski days, the beers on retreat nights and all the shared moments with junk food on the cold winter nights of SIGGRAPH's deadlines.

Finally, I would like to especially thank the amazing friends I've done during my stay in France: *Marta*, *Simona*, *Alfio*, *Rachid*, and *Anaïs*. I'm deeply grateful to have met you. You've been my support along these years so far away from home and I can't see a way of going through this PhD without you *¡Los voy a extrañar!* Last, but not least, I would like to thank my family, especially my parents *Manuel* and *Adriana* who always believed and supported me in ***every imaginable way***.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Problem statement and methodology 3

1.3 Contributions 5

1.3.1 Automated Drawing Guidance and Feedback from Photographs 5

1.3.2 Computer-Assisted Crafting of Wire Wrapped Jewelry . . 6

1.3.3 Regularized Curvature Fields from Rough Concept Sketches 7

1.4 Structure of the Thesis 8

2 Automated Drawing Guidance and Feedback from Photographs **9**

2.1 Introduction 9

2.2 Related Work 11

2.3 Design Goals 13

2.4 System Overview 14

2.5 Visual Guides 15

2.5.1 Laying Down the Main Drawing Structure 16

2.5.2 Drawing Contours and Details 18

2.5.3 Verifying Alignments and Proportions 18

2.6 Registering Visual Guides and User Input 21

2.6.1 Dense Correspondence between Contours 21

2.6.2 Sparse Correspondence between Corners 22

2.7 User Interface Design 23

2.7.1 Visual Guides and Drawing Layers 23

2.7.2 Corrective Feedback 24

2.7.3 On-Canvas vs. On-Model Guidance 25

2.8 Evaluation 26

2.8.1 Method 27

2.8.2 Results 27

2.9 Discussion 30

2.10 Conclusion 33

3 Computer-Assisted Crafting of Wire Wrapped Jewelry 35
3.1 Introduction . . . 35
3.2 Related Work . . . 38
3.3 Wire-Wrapping Principles . . . 39
3.4 Wire Decomposition . . . 40
3.4.1 Line-Drawing Vectorization . . . 40
3.4.2 Energy Formulation . . . 41
3.4.3 Optimization Method . . . 43
3.4.4 Pre- and Post-Processing . . . 46
3.5 Assisting Wire Bending . . . 47
3.6 Evaluation . . . 48
3.7 Conclusion . . . 54

4 Regularized Curvature Fields from Rough Concept Sketches 55
4.1 Introduction . . . 55
4.2 Related Work . . . 57
4.3 Overview . . . 62
4.4 Stroke Constraints . . . 64
4.5 Estimating Curvature Fields . . . 65
4.5.1 Motivation for the BendField Energy . . . 65
4.5.2 Formal Derivation of the BendField Energy From Properties of Curvature Lines and Fields . . . 67
4.5.3 Lifting the Cross Field to 3D. . . . 71
4.5.4 Algorithm Overview . . . 73
4.6 Non-Orthogonal 2D Cross Fields . . . 73
4.7 Relation to Previous Cross Field Approaches . . . 81
4.8 Additional Details . . . 82
4.9 Results and Evaluation . . . 85
4.10 Conclusion . . . 92

5 Conclusion and Future Work 95
5.1 Short term goals . . . 95
5.2 Long term goals . . . 97
5.3 Concluding remarks . . . 100

Bibliography 101

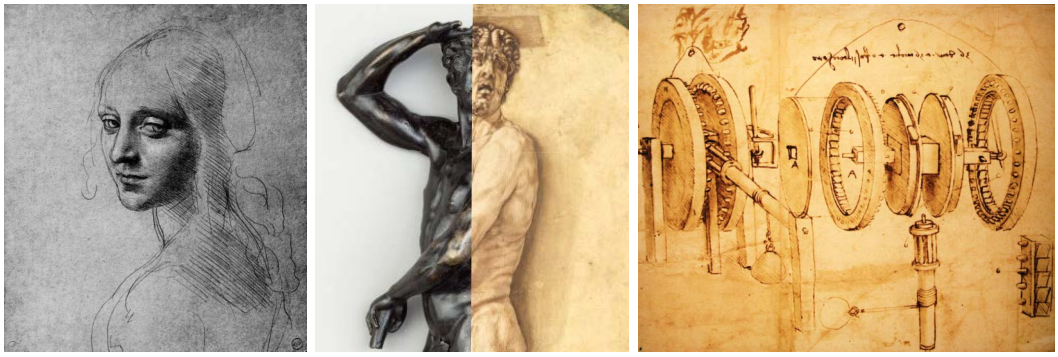
CHAPTER 1

Introduction

1.1 Motivation

Figure 1.1: Cro-Magnon artists painting in Font-de-Gaume, by Charles R. Knight

Drawing is the earliest form of visual depiction. Humans started to represent images with lines and colours before they invented writing. Evidence from around 40.000 years ago exists in many caves around the world, especially in Spain and France (Cueva de Altamira, Font-de-Gaume (Figure 1.1)). The first pictorial art represented animals and hunting scenes, painted with a very limited palette and using a cane or fingers as drawing tools. While these initial paintings are associated with religion or magical purposes, drawing developed over history to fulfill a variety of functions. From art to science, drawing has always been one of the most intuitive means of visual communication for human beings.



(a) Woman portrait by Leonardo Da Vinci

(b) Statue and inspiration sketch by Benvenuto Cellini

(c) Gears design by Leonardo Da Vinci

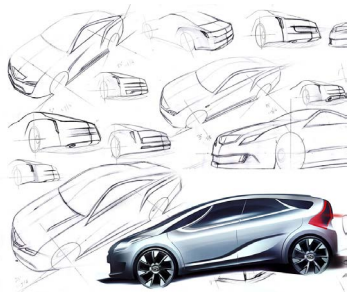
Figure 1.2: Pen and ink are versatile tools: Leonardo Da Vinci is able to express beauty on this sketchy portrait (a). Benvenuto Cellini worked on paper drafts and iterated multiple times before having a polished metal statue (b). Leonardo also drawed to describe the parts of a mechanical design based on gears (c).

In Europe, during The Middle Ages, drawing was used as an artistic tool to decorate books or represent religious scenes. Similarly in China, paintings and drawings depicted the life of powerful people and decorated the corridors of the imperial palaces. ***Drawing for art*** was mainly focused on the representation of realistic scenes using drawing from observation techniques (Figure 1.2a). The Renaissance brought new methods and an explosion of the art production. Given its strong creative power, drawing became a fundamental tool in other visual arts. Sculptors and architects relied on sketches as the first step to express ideas. ***Preparatory drawing*** was a common practice that allowed artists to freely plan a composition in two dimensions before sculpting in the physical world (Figure 1.2b). In addition to artistic applications, the use of drawing quickly extended to other disciplines. Particularly, ***drawing for engineering*** was exploited early by inventors and scientists to record and communicate their findings to others. Leonardo Da Vinci, for instance, relied on his drawings to document mechanical designs and studies on the human body (Figure 1.2c).

While the above usage of drawings was reserved to a restricted elite, the modern era opened drawing possibilities to a broader audience. Motivated by cheaper supplies and the joy in ***drawing for art***, an increasing number of people

(a) Drawing class (1918)

(b) Chrysoprase necklace sketch and fabricated piece



(c) Concept sketches for car design (Hyundai)

Figure 1.3: Drawing class at Minneapolis College of Art and Design, 1918 (a) (source: pixshark.com). Necklace concept and fabrication by a hobbyist (b) (source: examiner.com). Concept sketches in car design allow designers to communicate with 3D modellers (c) (source: hyundai-veloster.eu).

started to learn drawing techniques (Figure 1.3a). Additionally, in recent years, the democratisation of the means of production made design a popular activity. ***Preparatory drawing*** is now practiced by non professionals who use affordable processes to create and build their own goods (Figure 1.3b). Nevertheless, drawing still plays a predominant role in traditional industry. Mass production involves a large number of people at each stage of the pipeline, with very specialised skills and tasks. ***Drawing for engineering*** is a key communication tool between design and engineering stages in the production chain, allowing high quality standards (Figure 1.3c).

1.2 Problem statement and methodology

Our goal in this thesis is to facilitate and accelerate drawing for amateurs as well as for expert designers and illustrators, employing graphics, image processing and interaction techniques. Since this is a broad spectrum to tackle, we identify three specific problems related to drawing for art and engineering.

First, we want to help and encourage beginners to practice drawing for art (Chapter 2). The main challenges novices face are learning to observe a scene, properly set up the main elements in a composition, and use visual clues to guide

their strokes. We propose an interactive tool to guide them in this task. Second, we want to help hobbyist makers convert a preparatory drawing into a physical piece of jewelry (Chapter 3). Main challenges are decomposing an input drawing into a set of metal wires and bending the wire to give it shape. Finally, we help professional designers produce 3D renderings from their rough concept sketches (Chapter 4). These renderings traditionally support communication with engineers and clients. However, retrieving the information required to create these renderings is challenging due to the inherent 3D ambiguity of a 2D sketch and the roughness in the input drawing.

Our three contributions target different kinds of users: learners, hobbyists and professionals. However, inspired by related work on computational visual communication [Agrawala 2011], we have applied a common methodology to structure each of our projects, which consists of three main steps:

Understand how people achieve the tasks. We first study the literature on each artistic task: teaching books, online forums, tutorials and step-by-step videos. To complement them, we observe how professionals and skilled artists achieve good results. Formalising artistic practices from these sources raises several challenges. Techniques are illustrated with specific examples, with differences proper to the habits of each designer. Additionally, the techniques are discussed with an artistic vocabulary, which is not always precise enough to allow an algorithmic implementation. We confront many sources of information in order to identify common principles, discriminating between general practices and artist's personal style.

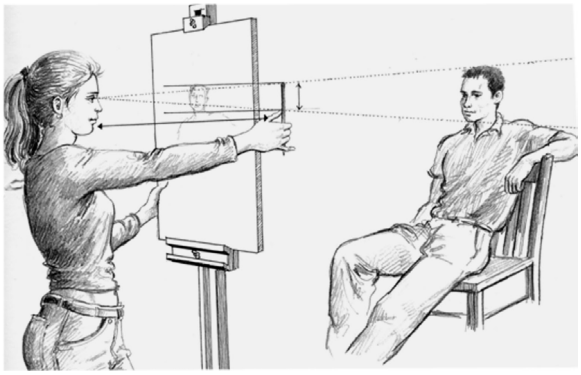
Assist and automate parts of the process. Once we have distilled the general principles, we translate them into algorithms that facilitate a given task. Our algorithms take as input an image, photographs or drawings, for which we want to extract some structure, i.e., drawing guidance, wire decomposition, 3D information. Because this input is often ambiguous, we use our principles to reduce the space of solutions. In Chapter 2 this translates into a small yet expressive set of guides that we extract from photographs, while in Chapters 3 and 4 we implement the principles as regularization terms in an optimisation guided by input drawings.

Evaluation. Finally, we evaluate the impact of the proposed tools. We conduct user studies and gather precise quantitative measurements of users performing tasks with and without the automatic assistance. We also ask users to provide direct feedback about the tools. Additionally, we compare our synthetic results to previous work in the field and to artist's existing designs.

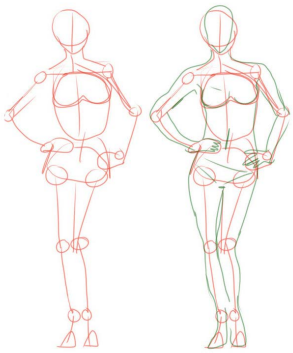
1.3 Contributions

We present three contributions that tackle drawing-related problems for learners, hobbyists, and professionals.

1.3.1 Automated Drawing Guidance and Feedback from Photographs



(a) Artist taking measurements using "sighting"



(b) Sketch scaffolding with skeleton

Figure 1.4: Traditional drawing from observation techniques are based on visual clues and measurements (a) (source: figuredrawingfall2014.wordpress.com). Observation can also be supported with construction lines (b) (source: deviantart.com).

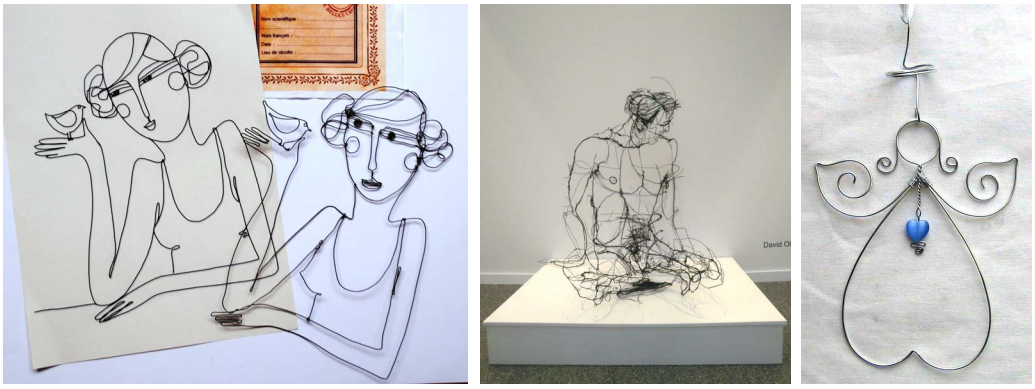
To assist ***drawing for art***, in Chapter 2 we present an interactive drawing tool to help novices practice drawing from observation using a photograph as a model (Figure 1.4).

Following our common methodology, we build this tool supported by the drawing literature, which describes a number of techniques to help people gain consciousness of the shapes in a scene and their relationships. We compile these

techniques and derive a set of construction lines that we automatically extract from a model photograph using state-of-the-art methods from Computer Vision. Our interface displays these lines over the model to guide its manual reproduction by the user on the drawing canvas. We augment the user experience by providing corrective feedback, which we generate by registering the user drawing with the model.

Our user studies show that automatically extracted construction lines can help users draw more accurately. Furthermore, users reported that guidance and corrective feedback help them better understand how to draw.

1.3.2 Computer-Assisted Crafting of Wire Wrapped Jewelry



(a) Metal wire 2D sketch

(b) Metal wire 3D sculpture

(c) Wire-wrapped jewelry piece

Figure 1.5: Wire-wrapping is a common exercise in design schools (a) (source: pinterest.com). Some artists even create sculptures by bending wire in space (b) (source: davidmigueloliveira.blogspot.pt). Hobbyists fabricate pieces of jewelry with metal wire and beadings (c) (source: http://tradslojd.ifokus.se/).

Wire wrapping is a traditional form of handmade jewelry that involves bending metal wire to create intricate shapes. This fabrication technique can be seen as a way of drawing in space with wire (Figure 1.5). The technique appeals to novices and casual crafters because of its low cost, accessibility, and unique aesthetic. With the goal of exploring the transition between ***preparatory drawing*** and fabrication, in Chapter 3 we present a computational design tool that addresses

the main challenges of creating 2D wire-wrapped jewelry. Our main contribution is an automatic wire decomposition algorithm that segments a drawing into a small number of wires based on aesthetic and fabrication principles. We formulate the task as a constrained graph labeling problem and present a stochastic optimization approach that produces good results for a variety of inputs. We then generate a 3D-printed custom support structure that helps users bend the wire into the appropriate shape.

We validate our wire decomposition algorithm against existing wire-wrapped designs, and use our end-to-end system to create new jewelry from clipart drawings. We also evaluate our approach with novice users, who were able to create various pieces of jewelry in less than half an hour.

1.3.3 Regularized Curvature Fields from Rough Concept Sketches

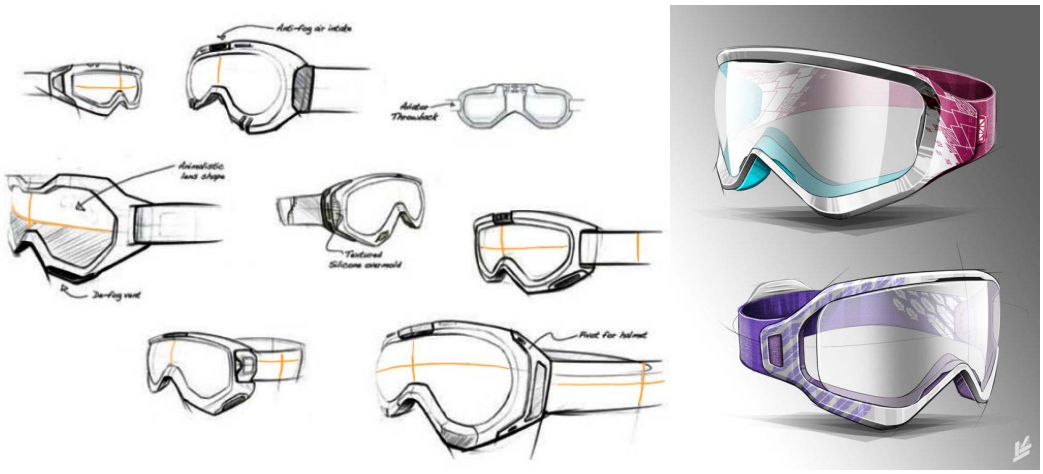


Figure 1.6: Designers shade and colour their sketches for presentation. In rough sketches, curvature lines convey bending of smooth surfaces (source: coroflot.com, behance.net).

When ***drawing for engineering***, designers communicate the intended 3D shape using shading and texturing. However, such polished drawings are time consuming and tedious to create. Professionals convey 3D more quickly in rough sketches by drawing curvature lines (Figure 1.6). In Chapter 4, we present a method to exploit this information and extrapolate a dense curvature field in a rough concept sketch. We cast this problem as a scattered data interpolation, which

allows us to recover the intended surface normal at each pixel of the sketch. We then use the 3D information to automate shading and texturing on top of the sketch.

We demonstrate our algorithm on a variety of concept sketches with various levels of sketchiness. This tolerance is important to directly apply the shading on rough sketches, without tedious tracing of vectorial curves.

1.4 Structure of the Thesis

We detail our contributions in the next three chapters. For each we follow our common methodology, first describing the domain-specific principles, then the algorithms we derived from these principles and finally our evaluation. Because of the diverse nature of the problems we address, we present specific related work within each chapter rather than as a separate one.

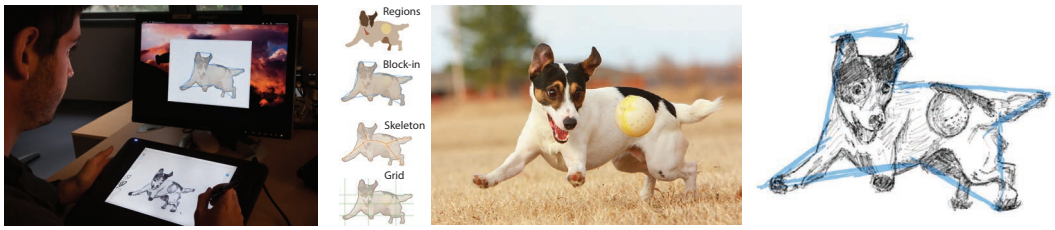
CHAPTER 2

Automated Drawing Guidance and Feedback from Photographs

2.1 Introduction

A major challenge in drawing from observation is to trust what we *see* rather than what we *know* [Nicolaides 1969; Edwards 1979; Dodson 1985]. Our mental image of common objects is iconic and conflicts with the particular instance that we observe, resulting in distorted or simplistic drawings [Eitz 2012]. Drawing books and tutorials provide simple techniques to help learners gain consciousness of shapes that they observe and their relationships [Edwards 1979; Dodson 1985; Hoddinott 2011; Bradley 2003; Hoddinott 2012; Kohr. 2012]. Common techniques include drawing simple geometrical shapes – also known as *blocking in* – before drawing the subject of interest and checking for alignments and equal proportions. While very effective, these techniques are illustrated on few examples with static instructions and no corrective feedback. As it takes significant effort to generalize the techniques to arbitrary models, books and tutorials benefit only few dedicated learners.

Interactive technology and pen-based interfaces offer new possibilities for the dissemination of drawing techniques to a larger audience by providing assistance and encouraging practice. Recent work in this area includes the iCanDraw? system to draw faces [Dixon 2010] and ShadowDraw that suggests completion of the drawing as it is performed [Lee 2011]. Following this line of work, we present an interactive drawing tool that assists users in the practice of long-standing drawing techniques advocated by expert teachers. Our *drawing assistant* helps users to practice these techniques from any model photograph and provides corrective feedback interactively. From a technical standpoint, we make two primary



(a) Interaction setup (b) Model and extracted guides (c) User construction lines and drawing

Figure 2.1: Our drawing assistant provides guidance and feedback over a model photograph that the user reproduces on a virtual canvas (a). We use computer vision algorithms to extract visual guides that enhance the geometric structures in the image (b). In this example, the user first sketched the block-in construction lines (c, blue) before drawing the regions and adding details. This guidance helps users produce more accurate drawings.

contributions:

Automatic visual guides to support drawing. Books and online tutorials on drawing abound with techniques that help people to observe shapes and draw them accurately. Many of these techniques share common principles implemented in different variations. We distill these principles and derive a set of visual guides, in the form of construction lines that assist users in the process of drawing from a model. We extract visual guides automatically from photographs using simple computer vision algorithms.

Interactive drawing tool with corrective feedback. We integrate our visual guides into a drawing tool that displays guidance over a model photograph. Following recommendation from the drawing literature, we separate our interface into two display areas — model and canvas, as shown in Figure 2.1 — to encourage users to concentrate on the subject of the drawing rather than on the drawing itself. This interface design differs from existing methods that provide guidance directly on canvas.

Our pen-based interface also allows users to navigate between the techniques they wish to practice and to draw construction lines in dedicated layers. We augment this interface with real-time registration between the drawing and the

model to detect errors and distortions. We then provide corrective feedback by highlighting in the model alignments and equal proportions that are violated in the drawing.

We conducted two user studies to inform the design of our tool and evaluate our approach with a total of 20 users. Participants produced better drawings using the drawing assistant, with more accurate proportions and alignments. They also perceived that guidance and corrective feedback helped them better understand how to draw. Finally, some participants spontaneously applied the techniques when asked to draw without our tool after using it for about 30 minutes.

2.2 Related Work

On-Canvas Guidance. Several systems assist the process of drawing by displaying guidance on the drawing surface. Projector-Guided Painting [Flagg 2006] decomposes a target painting into coarse-to-fine layers. Users paint over the projection of each layer, following guidance to orient individual brush strokes or to paint all the strokes of a given color. Similarly, PapARt [Laviole 2012] allows users to trace over a 3D scene projected on paper and Rivers et al. [Rivers 2012] extend this concept to project guidance for sculpture. ShadowDraw [Lee 2011] provides guidance for freeform drawing by inferring potential models from the user sketch. At run time, the algorithm matches the drawing to a database of images and blends the best matches to form a *shadow* that suggests a completion of the sketch to users. Similarly, Limpaecher et al. [Limpaecher 2013] correct sketches traced over a picture by matching them against a database of drawings of the same picture.

All these methods are reminiscent of the traditional "paint-by-number" and "connect the dots" drawing books that guide people in placing individual strokes until completing complex artworks. While these approaches can give people confidence in their ability to draw, they do not help them observe and understand the underlying shapes, relationships and proportions of the drawn models.

Step-by-Step Instructions. A complex drawing can be easier to achieve if it is decomposed into a succession of simple steps. Several commercial

applications propose step-by-step drawing tutorials, such as Nintendo Art Academy[1] and the "How to Draw" and "Learn to Draw" mobile applications. Unfortunately, these tools do not provide any corrective feedback to the inexperienced user. Sketch-Sketch Revolution [Fernquist 2011] allows expert users of sketching software to generate tutorials for novice users. The system offers on-canvas guidance and feedback to replicate the expert strokes at each step of the tutorial. Finally, work in other domains has introduced systems that generate tutorials from demonstration for image editing [Grabler 2009; Chi 2012] and 3D modeling [Denning 2011]. Such tutorials illustrate drawing techniques on pre-recorded examples rather than images of the user's choice.

Closer to our work are the iCanDraw? and EyeSeeYou systems [Dixon 2010; Cummmings 2012] that assist users in drawing faces and eyes respectively. These systems rely on face and sketch recognition algorithms to generate domain-specific instructions and textual or on-canvas feedback. We draw inspiration from these approaches, incorporating some of their design principles. However, the drawing assistant presented on this Thesis implements a different set of guides to draw arbitrary models. We also provide visual feedback that highlights alignments and proportions on the model photograph, helping people to see and correct the relationships between different parts of a shape. Our approach also draws inspiration from the system described by Soga et al. [Soga 2009], which guides users to draw a still life scene using a data base of pre-recorded construction lines and advice.

3D Sketching. While our primary goal is to assist users in drawing from photographs, we believe that helping people to draw has the potential to benefit other applications that rely on sketches as input, such as sketch-based modeling. Several systems use construction lines [Schmidt 2009b] and sketching planes [Bae 2009] to support 3D curve sketching. However, users of these systems create construction lines from imagination rather than from a model.

[1] http://artacademy.nintendo.com/

2.3 Design Goals

While most children enjoy drawing, many adults consider themselves incapable of drawing realistically and resort instead to iconic sketches of objects [Eitz 2012]. Edwards [Edwards 1979] suggests that people confront a frustrating *artistic crisis* around ten as their abstraction of the world — what they *know* — conflicts with their visual perception — what they *see*. For instance, children commonly draw cubes with squared faces and get disappointed by the result, as it bears unrealistic proportions and lacks perspective. To gain confidence and improve their drawing skills, people need to resolve this conflict and focus on the actual forms that they want to draw rather than their symbolic representations: "You should set your symbol system aside and accurately draw what you see" [Edwards 1979]; "We should draw as if we know nothing, and were obedient only to what our eye tells us to draw" [Dodson 1985]. Dodson [Dodson 1985] also observes that "a common practice that weakens drawing effectiveness is concentrating too much on your paper and not enough on your subject". For this reason, art teachers advise students to "keep their eyes on the subject of the drawing most of the time, not on the drawing itself" [Nicolaides 1969; Edwards 1979].

Inspired by these recommendations, we set the following design goals:

- Encourage users to focus their attention on the actual model rather than their drawing.
- Help users to practice *observation techniques* proposed by the drawing literature. These techniques should allow users to identify the shapes and their relationships on a model and to structure their drawings.
- Support corrective feedback to help users understand their errors and refine their drawings.

In addition to our three design goals, we also chose to focus on basic drawing techniques that apply to generic models, rather than domain-specific rules such as anatomy and perspective. This choice is motivated by the teaching approach of Dodson [Dodson 1985], who states that domain-specific principles "were developed to help us understand what we see, but they do not come first. Seeing comes

first. When rules conflict with seeing, forget them and draw what you see."

2.4 System Overview

Figure 2.2a illustrates the main interface of our interactive drawing assistant. It consists of two distinct areas that we display on two separate monitors. The *model* area shows the photograph, which acts as the model for the drawing task, while the *canvas* is the drawing area where the user interacts with the pen. We display the model on a vertical computer monitor and the canvas on a pen display, which mimics traditional drawing where the drawer alternates between observing the model and drawing on paper.

Given a model photograph that the user wishes to reproduce, we first run computer vision algorithms to extract visual guides that emphasize the shapes in

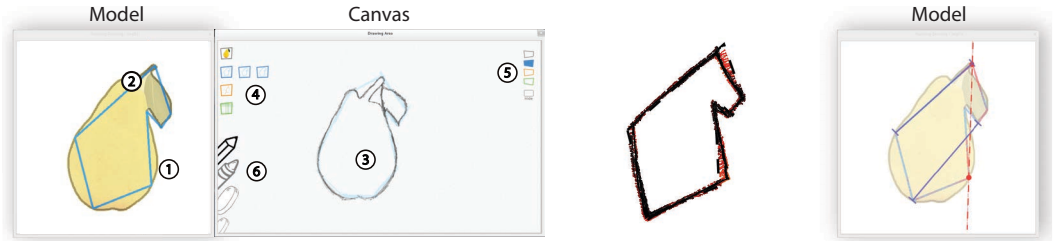


Figure 2.2: Our interface is composed of two display areas (a): the model area with the photograph and the visual guides, and the canvas area with the tools and the user's drawing. The user has used the drop-down list of tools (4) to activate a coarse block-in guide. The block-in guide is displayed over the model in blue (2). The user has reproduced the block-in guide over the canvas in the corresponding blue layer (5) and used these construction lines as a scaffold to reproduce a detailed contour (1,3). We offer simple drawing tools including a pencil, a pen and a small and big eraser (6). Our system registers the drawing in the active layer — block-in in this example — to estimate distortions (b) and shows on the model the erroneous alignments and proportions (c). In this example, the red dashed line shows a vertical alignment that has not been respected by the user and the dark blue segments show two distances that should be made equal.

the image. We derive these guides from traditional drawing techniques that we discuss in the following section. Our interface displays the detected guides over the model, and the user reproduces them on the canvas as construction lines that form a scaffold for more detailed contours.

Following our first and second design goals, we display visual guides in the model area to enforce users to observe the model before structuring their drawing by themselves. While this design requires extra effort from the user and can be less precise than on-canvas guidance, it is consistent with the teaching approach and recommendations of expert artists.

At run time, our system registers the construction lines drawn by the user with the corresponding visual guides (Figure 2.2b). We use this registration to estimate local distortions and to detect erroneous alignments and proportions in the drawing. The user can then ask for feedback based on this evaluation. The feedback highlights the parts that require extra attention over the model (Figure 2.2c) so that the user can understand what to observe to improve the drawing.

2.5 Visual Guides

Drawing books [Edwards 1979; Dodson 1985; Bradley 2003; Hoddinott 2011] and online tutorials [Hoddinott 2012; Kohr. 2012] abound with recommendations to observe the shapes in a scene and identify their relationships. While most authors only present a subset of techniques and describe them in their own vocabulary and style, we distilled from these resources three main principles suitable for integration in a computer-assisted tool:

- Drawers should first lay down the main structure of the drawing with a coarse approximation of the shape.
- The coarse structure forms a scaffold to guide contour drawing. Drawers should draw contours of large regions first and then details.
- Proportions and alignments should be verified to avoid distortions.

We refined these principles through informal user tests both on paper and the

computer.

We articulate each principle around *visual guides* that help users construct their drawing. We then describe how to extract these visual guides from a photograph using existing computer vision algorithms. Our guides do not aim to match the style of a particular artist but rather to capture the common idea of drawing from coarse to fine. Art books also describe drawing as "a process which usually bypasses conscious thought and knowledge" [Dodson 1985] and we found simple vision algorithms that are blind to the semantical content of the image, to be very effective to help the user "bypass" advanced cognitive processes[2]. To further simplify our analysis, we assume that users first separate the subject of the photograph from its background using an interactive foreground extraction tool like *GrabCut* [Rother 2004].

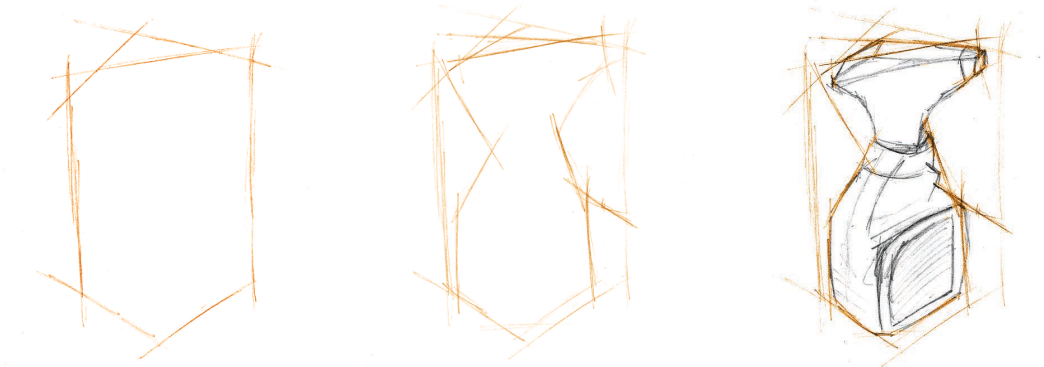
2.5.1 Laying Down the Main Drawing Structure

Inexperienced drawers often strive to know how to start their drawing. A common recommendation is to first sketch the basic structure of the shape that will then serve as construction lines to support more complex details.

The *block-in* technique approximates the shape with a polygon [Kohr. 2012] or with a collection of geometrical primitives like disks and rectangles [Hoddinott 2011]. We experimented with both approaches and our early tests with users revealed that using multiple disks and rectangles quickly produces cluttered visual guides that intersect inside the shape. We adopt instead the polygonal visual guide that artists often only apply on the main outline of an object to avoid clutter. This approach draws inspiration from sculptors who start with a block of wood or marble and remove matter from coarse to fine until they reach the final shape (Figure 2.3a).

Artists also use skeletons to enhance the structure of a shape [Bradley 2003]. While the block-in technique emphasizes the outer shell of man-made objects, skeleton lines depict the principal internal directions and are more suitable to elongated structures and characters, even though they do not necessarily correspond to an anatomical skeleton (Figure 2.4a).

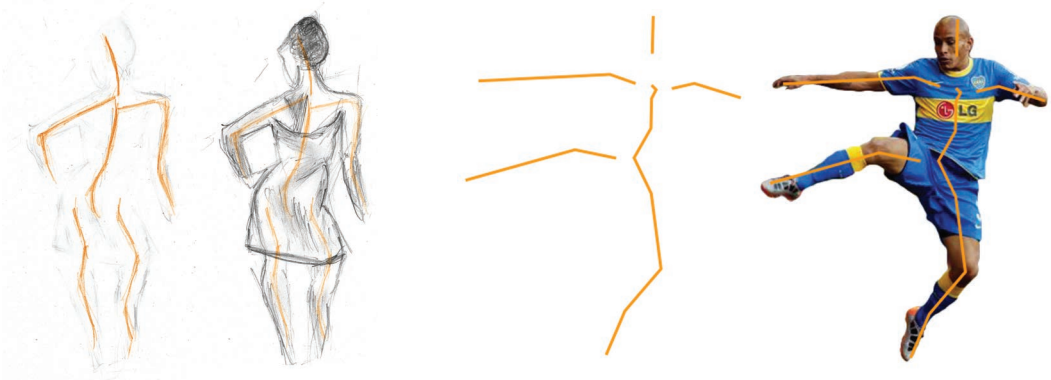
[2] We provide the results of our visual guide extraction on 10 typical images as supplemental materials.



(a) Hand-drawn block-in lines

(b) Automatic extraction

Figure 2.3: Blocking-in consists in first drawing a coarse approximation of the shape before adding details (after [Kohr. 2012]).



(a) Hand-drawn skeleton

(b) Automatic extraction

Figure 2.4: In this example, skeleton lines enhance the main directions of the body parts. Note that the lines do not correspond to an accurate representation of an anatomical skeleton (after [Bradley 2003]).

Automatic extraction Many algorithms could be used to generate a polygonal approximation of an object's outline. The only requirement is to preserve the overall shape in order to provide local guidance that prevents drifting away while drawing detailed contours. We use the *Douglas-Peucker* algorithm [Douglas 1973], which simplifies a shape by progressively removing vertices of its contour. We generate a coarse-to-fine approximation with two levels of detail containing seven and ten vertices (Figure 2.3b).

To extract the skeleton of an object, we use the medial axis algorithm [Blum 1967] which generates the set of points having more than one closest point on the contour of a region. However, we found the medial axis of the main outline of an object to be too detailed. Instead, we first approximate the outline with the detailed polygon of the block-in guide and then compute its medial axis to obtain a skeleton composed of few line segments (Figure 2.4b).

2.5.2 Drawing Contours and Details

The block-in and skeleton guides form a coarse scaffold for drawing the detailed contours of color regions. Edwards [Edwards 1979] and Dodson [Dodson 1985] advise to consider these regions as abstract shapes that compose a "jigsaw puzzle". They recommend to draw the large regions first, then the smaller ones, merging adjacent regions that share similar tones (Figure 2.5a). Focusing on the abstract shapes of individual regions prevents us from thinking about the semantic of the object that the regions compose.

Automatic extraction We guide users in identifying large and small regions by segmenting the image with the hierarchical algorithm of Arbelaez et al. [Arbelaez 2011] (Figure 2.5b). We chose this algorithm for its high score on the Berkeley Segmentation Dataset Benchmark [Martin 2001] with respect to human ground-truth boundaries. We generate two levels of detail containing one and six regions.

2.5.3 Verifying Alignments and Proportions

It is often hard to judge and measure the distortions in a drawing with a naked eye. Artists make use of the "sight" (or "thumb-and-pencil") method to facilitate this task. They hold their pen or pencil at arms length between their eye and the

(a) Hand-drawn regions

(b) Automatic extraction

Figure 2.5: Considering a shape as a collection of regions helps to forget its iconic representation (after [Dodson 1985]).

object of interest and sight along it. Sighting helps them identify and estimate relative proportions and alignments [Dodson 1985] (Figure 2.6). While not explicitly stated in drawing books, we observed that artists only look for vertical and horizontal alignments, which are easier to reproduce. Sighting is especially useful when beginning a drawing to obtain well proportioned construction lines that then yield accurate contours.

Artists sometimes prepare their drawing by laying down a grid over the drawing surface [Bradley 2003; Hoddinott 2011]. The grid serves multiple purposes, such as visualizing alignments and spacing and helping the artist focus on local areas in the drawing. The grid also serves as an alternative to block-in to capture the overall arrangement of shapes. Bradley [Bradley 2003] recommends to capture the most salient points of the subject with a non-uniform grid.

Automatic extraction To emulate sighting, we first detect feature points that correspond to salient landmarks in the image. Many feature detectors could be used for this task and we found that the Shi-Tomasi corner detector [Shi 1994] performs

Figure 2.6: Sighting facilitates the identification of alignments and proportions (after [Dodson 1985]).

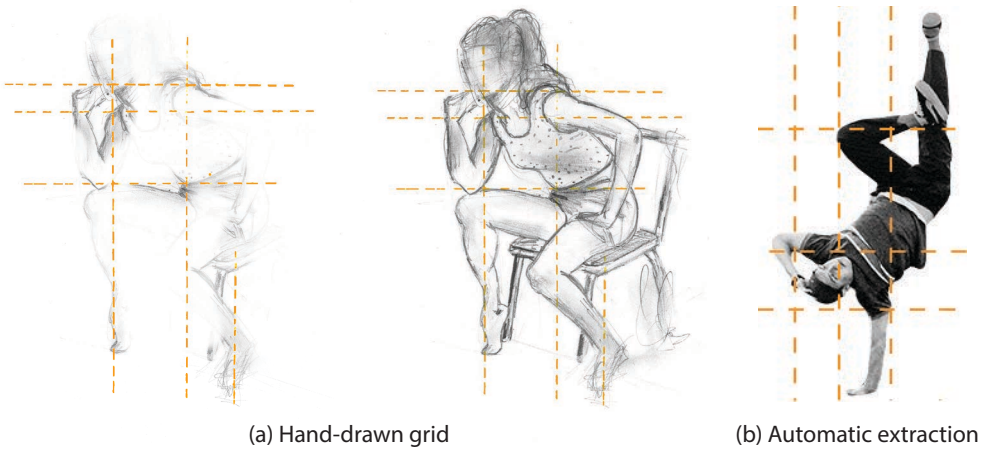


Figure 2.7: A grid visualizes alignments and spacing and allows artists to focus on one cell at a time (after [Bradley 2003]).

well for our purpose, both on photographs and line drawings. We then detect and sort pairs of aligned points. We favor pairs that form long lines close to vertical or horizontal because our first tests revealed that they were the most useful to prevent large distortions. For proportions, we detect points that form pairs of equal length. Following early comments from an expert user, we favor pairs that have a common point because users can more easily compare them using their pen as a compass. Finally, we build a non-uniform grid over the photograph by tracing vertical and horizontal lines through the n most salient corners ($n = 3$ in our implementation).

2.6 Registering Visual Guides and User Input

A key ingredient of our drawing assistant is the ability to evaluate error in the drawing with respect to the visual guidance. We perform this evaluation by registering each visual guide with its user-drawn counterpart in real time. We encourage users to draw different guides in different layers to facilitate registration, as we describe in the next section. Our registration builds both a dense correspondence between contours and a sparse correspondence between corners, which we use to provide different types of feedback.

2.6.1 Dense Correspondence between Contours

We first compute a dense correspondence between the guide and the user drawing to evaluate error along any portion of a contour. We base our registration on the *Shape Context* descriptor [Belongie 2002], which was designed for this purpose.

Shape Context represents a shape as a dense set of points where the descriptor of each point encodes the distribution of all other points relative to it. To register two shapes, the original algorithm first computes a one-to-one assignment between their respective point sets, and then estimates the transformation that best align the two shapes. Since the one-to-one assignment is a costly procedure, we adopt a faster approach and simply assign each point p of one shape to the most similar point q in the other shape. We use this assignment to estimate the affine transformation T_A between the two shapes:

$$\underset{T_A}{\operatorname{argmin}} \sum_{p} \|T_A(p) - q\|^2. \tag{2.1}$$

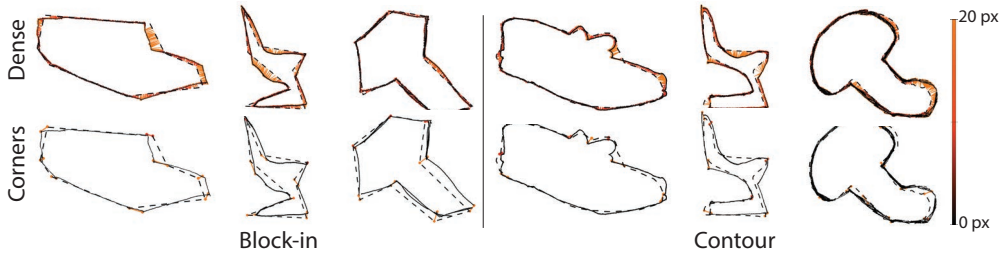


Figure 2.8: Results of our dense and sparse registration on drawings performed by three different users.

In practice, we always register the user drawing with respect to the guide so that each point in the drawing has a correspondence. Belongie et al. [Belongie 2002] suggest iterating the assignment and estimation steps to reject outliers from the initial assignment. We perform one such iteration, where we weight the matching costs $C(p, q)$ between a point and all other points by their residual after the initial transformation:

$$C'(p, q) = \|T_A(p) - q\| \, C(p, q). \tag{2.2}$$

The affine transformation yields a registration robust to non-uniform scaling and shear. While this robustness is desirable for identifying accurate correspondences, we should account for the non-uniform distortions when evaluating the quality of the drawing. For this reason, we also estimate the *similarity* transformation T_S that best approximates the registration as a combination of rotation, translation and uniform scaling. We express the error ε for each point in the drawing as the distance between the transformed point and its correspondence:

$$\varepsilon(p) = \|T_S(p) - q\|. \tag{2.3}$$

2.6.2 Sparse Correspondence between Corners

Our system relies on corner features to detect alignments and equal proportions. We register corners between the guidance and the drawing to verify that the user sketch satisfies the same alignments and proportions as the visual guide. We again use the Shape Context descriptor to perform this registration, using the affine transformation T_A computed from the dense correspondence to weight the matching cost of each pair of corners. In practice we use the inverse of this transformation since our goal is to find a correspondence between each corner of

the guide and its most similar corner in the drawing. While we need to run the Shi-Tomasi detector [Shi 1994] to extract corners from the user sketch and the region guide, we can directly use the vertices of the block-in and skeleton lines as robust corners for these guides.

We illustrate in Figure 2.8 our dense and sparse correspondence on block-in guides and contours drawn by three different users. We provide additional registration examples on other guides and images as supplemental materials.

2.7 User Interface Design

Drawing books do not provide clear recommendations about which guides to use for a certain subject or how to combine different guides together. A guide can be more appropriate for a given subject depending on its form and complexity, while the experience and style of the user may also determine the use of a technique. Therefore, we offer users the freedom to experiment with different guides on the same model and decide by themselves which technique best suits their needs.

2.7.1 Visual Guides and Drawing Layers

The canvas area offers simple drawing tools — a pen to draw opaque strokes, a pencil to draw light strokes, a small and a big eraser. The left side of the canvas contains a menu to select and configure the techniques users wish to practice, which includes regions, block-in, skeleton and grid. Each technique is associated with a different color and each icon displays a miniature visualization of its effect on the model (Figure 2.2a(4)). A drop-down list gives access to levels of detail for the block-in tool (coarse and fine polygon) and the regions tool (coarse and fine regions, and original photograph). We make the block-in and skeleton techniques mutually exclusive because they have a similar goal and displaying them together produces clutter. In contrast, we always show the regions behind other guides as they form the basic elements of the final drawing. Finally, the grid is optional and can be combined with any other technique.

We enrich our drawing tool with a layering system that has two benefits. First, it helps users make the distinction between the guides that they sketch and the final

drawing. Users can also hide the layers to visualize their drawing without construction lines. The second goal of the layering is to facilitate registration between the visual guides that we extract from the photograph and the guides drawn by the user. We assign one layer to each guide and compute a registration between each layer and the corresponding guide. We further help users to distinguish each layer by using different colors that correspond to the colors of the guides (Figure 2.2a(5)). When the user selects a guide, the system automatically activates the corresponding layer. The user can also select a layer and the system activates the corresponding guide. We then render the active layer and active guide on top of the other ones. Finally, we use semi-transparent strokes for the guidance layers to allow users see their drawing through the construction lines and subtle animated blending to communicate transitions between different guides.

2.7.2 Corrective Feedback

At any time, users can request feedback on their drawing by pressing a button on the pen tablet. We again follow our design goals and display feedback on the model to encourage users to observe it more carefully before applying corrections on canvas. We use the correspondence between the visual guides and the user drawing to generate two types of corrective feedback, as we illustrate in Figure 2.2c.

We first use our dense measure of error to highlight parts of the guides that are highly distorted. We use a color code that interpolates between the color of the guide (no error) and red (high error). We then use the registration between corners to display the alignments and relative proportions that the user drawing does not satisfy. This form of feedback mimics the traditional sighting technique. Dashed lines indicate alignments while pairs of colored lines indicate equal proportions. We adjust the opacity of these indications proportionally to the magnitude of the error, so that users can focus on the most erroneous parts and assess their progress as the indications become more transparent until disappearance.

We observed in a first version of our tool that the model can get highly cluttered in the presence of multiple indications of equal proportions. We avoid such clutter by first displaying the indications that share a common point, since those are easier to verify by the user. We also disable the proportion feedback when the region tool

is active because the detailed contours yield too many candidate pairs of corners with equal length.

2.7.3 On-Canvas vs. On-Model Guidance

An important decision that we faced during the design process was whether to show guides not only over the model but also on the canvas. While we expected that on-canvas guidance would result in higher-quality drawings with less effort, we were afraid that it would distract users from observing the model, which would be contrary to our first design goal.

We conducted a pilot experiment to better understand the strengths and weaknesses of each design approach. Twelve participants (seven women and five men) tested three drawing interfaces that provided different levels of guidance: (I1) no guidance, (I2) guidance over the model, and (I3) guidance over the model and the canvas.

The use of corrective feedback was only applicable for I2. For each user interface, participants completed one practice task and two drawing tasks where they had to draw familiar objects from photographs. The order of presentation of the three interfaces was counterbalanced among participants. We kept the sessions short by focusing on the block-in technique and by limiting each drawing task to only 5 minutes so that the entire study took around an hour to complete.

As we expected, I3 resulted in better drawings, reducing contour error by an average of 50% compared to the base user interface (I1). In contrast, I2 did not have any immediate benefits for such short drawing tasks. However, participants appreciated that I2 encouraged them to observe the model and replicate the visual guides on the canvas by themselves.

One commented that I2 *"is the most didactic solution for learning how to draw"*, while I3 *"is the most easy solution but people do not learn"*. A second participant agreed that *"to learn how to draw, I prefer the [I2] tool. It is convenient to have the corrections"* while another said *"I liked to do the block-in by myself with the [I1] tool"* but added that *"if I would have started by [I1], I would not*

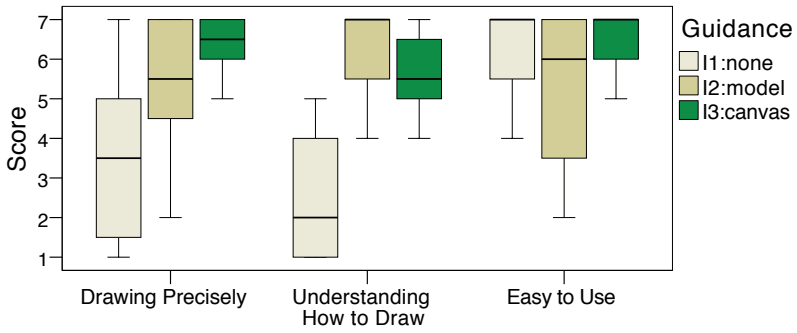


Figure 2.9: Pilot experiment: Boxplot summarizing the subjective user rankings (1 = not helpful, 7 = very helpful). The top and bottom of the box correspond to the first and third quartiles and the band inside corresponds to the median. The whiskers denote the score range.

have known how to draw the block-in guides". Participants also appreciated the use of corrective feedback: *"It is nice to have the feedback in [I2]. [I3] is good to draw fast."*

On the other hand, several participants liked the direct on-canvas guidance, and, as shown in Figure 2.9, they rated I3 relatively high. The tradeoffs among the three interfaces are perhaps best summarized by the following user comment:

"The three versions let me see the difference. It's the union of them that lets me understand the technique."

We decided to keep our tool simple and consistent with our initial goals by only showing guides over the model. However, we envision that future designs of the tool could accommodate all three levels of guidance within the same interface.

2.8 Evaluation

We conducted a user study to evaluate the automatic guidance of our drawing assistant and get initial user feedback about its potential as a learning tool.

2.8.1 Method

Participants Eight volunteers participated in the study — six women and two men, 23 to 43 years old. Their drawing experience ranged from 1 to 4 on a Likert scale from 1 (poor) to 5 (good), with a mean experience of 2.25 (below average). All the participants were right-handed.

Apparatus Participants interacted with a Wacom Cintiq 12WX pen display. Model photographs were displayed on a 21" monitor.

Design and Procedure Before each session, participants completed a brief questionnaire about their drawing experience. Each participant was then exposed to two versions of the drawing interface: our drawing assistant with on-model guidance and a base interface with no guidance. The order of their presentation was counterbalanced among participants, i.e., four participants started with the base interface, and four participants started with the guided one. The base user interface did not provide any mechanism for drawing on separate layers and we gave no instructions whether participants should make use of construction lines for this condition.

Participants were first given a short tutorial for each interface. Then, they completed a practice and a main drawing task that lasted 15 to 30 minutes. To observe the use of visual guides with different types of drawing subjects, we split participants into two groups (see Figure 2.12). The first group drew a roller skate and a trumpet and was advised to use the block-in guides. The second group drew two full-body characters and was advised to use the skeleton guides. We provide example sessions of each group in the accompanying video.

After the end of the session, participants completed a questionnaire to evaluate their experience with the tool. The whole procedure lasted 60 to 80 minutes.

2.8.2 Results

Figure 2.12 presents the drawings of each participant along with their mean contour error. This error is calculated by our similarity registration and provides an objective measure of the overall distortion of the drawing that we aim to

correct for. Since some participants drew more detailed drawings than others, we performed a fair comparison by manually erasing any interior contours and measuring error on the main outline of the drawn subject only.

The average error was 24.6 pixels ($SD = 9.0$ pixels) for the base drawing interface and 13.2 pixels ($SD = 4.6$ pixels) for the guided one. Guidance resulted in error reduction ranging from 31% to 64% for all but Participant 5. This participant, who reported having previous training in drawing, explained:

"The [skeleton] guide is very clear and I understood quickly what I was supposed to do [but] following the guide was hard for me because I'm used to drawing in a different way. I think this is very useful for a beginner."

Our system was particularly effective for participants 1, 6, and 7 that had poor drawing experience. A close examination of their drawings without guidance reveals significant errors. For example, Participant 1 made the roller shoe too tall and the wheels too close apart and Participant 6 made the torso of the character too long, the right leg too short and the left leg too low. No such distortions appear in their guided drawing.

Figure 2.10 presents how participants evaluated the visual guides, the corrective feedback, and their overall experience of the tool's learning utility.

Guides: Overall, participants found the visual guides clear, helpful, and easy to use. Seven participants reported having made extensive use of the visual guides. Only Participant 5 reported that she made limited use.

Corrective feedback: While most participants appreciated the corrective feedback, their ratings varied from $Neutral$ to $Good$. An analysis of the error evolution during the guided tasks reveals that by following the corrective feedback, Participants 1 to 7 managed to progressively reduce the error of their guides by an average of 34% ($SD = 9\%$). However, Participant 8 increased the error of the skeleton by 43% between his first and last request for feedback. The participant noted that *"trying to correct some specific parts sometimes makes other*

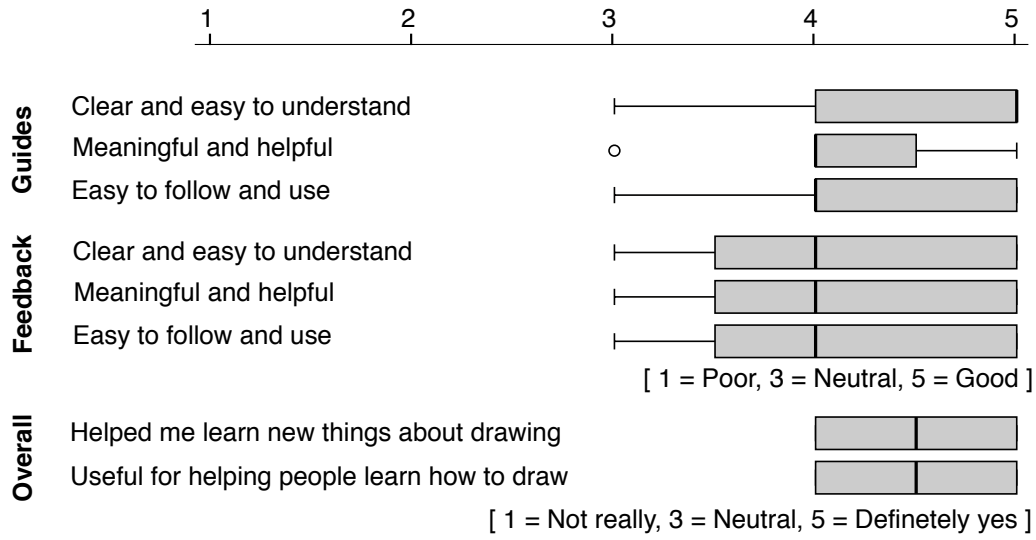


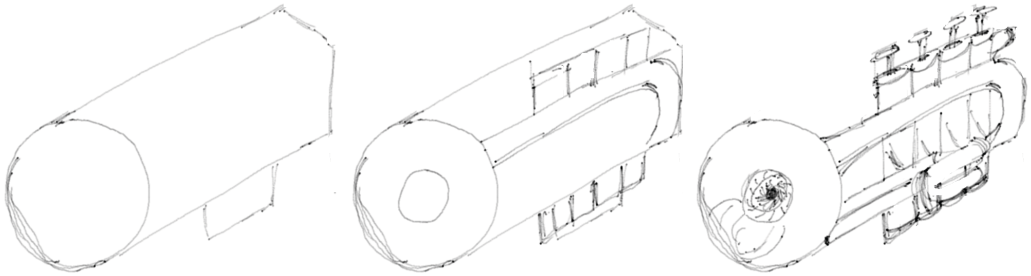
Figure 2.10: Subjective user evaluation of the drawing assistant. The dot represents an outlier.

proportions wrong". We suspect that skeletons were harder to draw because they contained around 15 vertices forming disconnected segments, while the block-in guides form closed polygons of no more than 10 vertices.

Overall experience: All the participants, to a different extent each, agreed that the session helped them learn new things about how to draw: *"I discovered new concepts like regions and block-in. I learned a fairly easy technique to improve my drawings."* Similarly, they agreed that the interface can be useful for helping people learn how to draw.

Interestingly, participants who were first exposed to our drawing tool tried to apply the techniques practiced through the first task to the second non-guided task, despite the fact that the base interface did not encourage their use. Participant 6 drew a skeleton to structure her second drawing. Participant 8 used sighting to verify alignments and proportions. Participant 4 applied the sighting and block-in techniques (see Figure 2.11). According to this participant:

"I could apply the methods on my second drawing, and I think they were very useful to better reproduce the photo. I understood clearly the interest of the



(a) User's block-in lines (b) Final drawing

Figure 2.11: Use of the block-in technique by Participant 4 for her second non-guided task.

explained method."

This result is particularly encouraging as it shows that some users could quickly benefit from our tool.

Finally, some participants identified limitations and proposed areas for improvement. A participant observed that the *"first stages of the process provided a lot of help for learning to draw the volumes with right proportions"* but commented that she *"had some problems adding details"* as this stage of drawing was *"less assisted"*. Other participants pointed to the lack of support for different drawing habits, especially assistance adapted to more experienced drawers. Participants from an early informal study also suggested providing guidance not only for drawing shape but also for shading. Furthermore, some participants tended to mimic the clean style of our guides even though our registration is robust to more sketchy lines. They suggested that rendering guides in a sketchy style would have made them feel more relaxed.

2.9 Discussion

While we have selected simple and robust computer vision algorithms to extract our visual guides, they may fail to detect the desired features on some images. As a result, our system can sometimes miss alignments or proportions that could help users improve their drawing. Alignments and proportions alone are also sometimes not enough to show how to improve a drawing, as is the case in

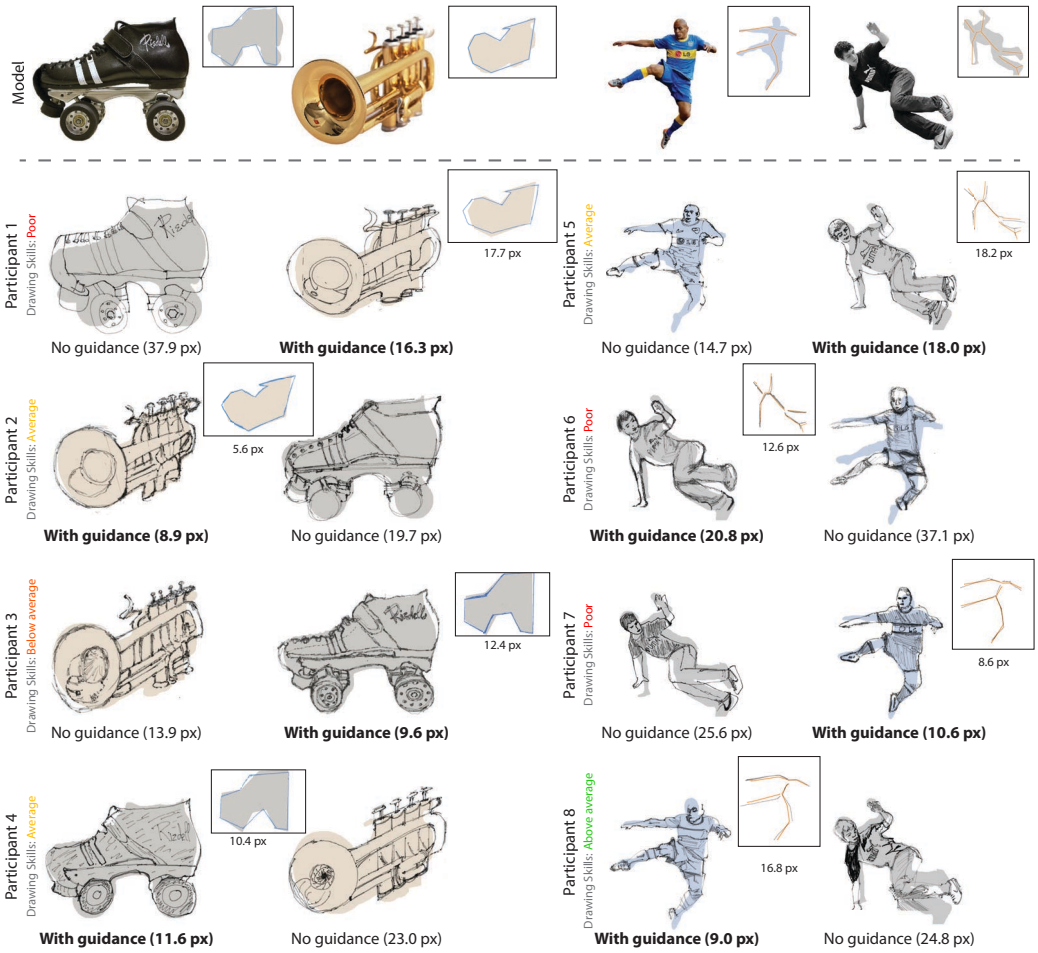
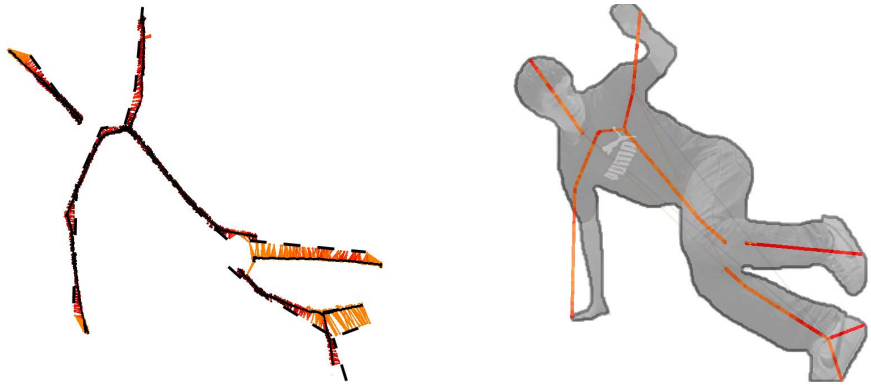


Figure 2.12: Drawings produced by the height participants, with and without our tool, in the order of completion. We provide as inset the drawn construction lines for the drawings performed with our tool. We display under each drawing the average error of the main contour, in pixels.

Figure 2.13 where our system did not identify any sighting guides to help the user move the legs apart. Traditional artists face the same limitations of sighting and resort to additional measurement techniques, which we could also integrate into our system. For example, some artists hold two pens as a compass to measure angles.

At several stages of the design process, we made the choice to avoid visual clutter by limiting the amount of guidance. For example, we chose to not show



(a) Registered drawing (b) Feedback

Figure 2.13: Limitation of our feedback. While the user drew most of the skeleton accurately, he did not manage to correct the legs that should be moved apart (a). Our feedback mechanism marked the distorted legs in red but did not suggest any relevant alignments and proportions (b).

all the detected proportions at once and only use block-in on the main outline, not on the small internal regions. Despite these efforts, the feedback can be visually complex in the presence of many errors and we plan to explore alternative visualizations and interaction techniques to improve the balance between level of guidance and clarity. Nevertheless, participants judged our visualization clear and easy to understand overall and managed to apply the system's advice to correct their drawings.

Our system relies on well-known observational drawing techniques from the extensive drawing instruction literature. Assessing how people learn from these techniques represents an exciting but difficult research challenge. Such a study would have to observe users over a long period to measure significant learning behavior but also deal with unintended learning effects between conditions. Our study evaluates instead the quality of the produced drawings, which demonstrates that guidance and feedback have immediate benefits. Nevertheless, the users' subjective evaluation and the tendency to reproduce guides in the no guidance condition suggests that our system helps understanding how to draw.

2.10 Conclusion

We presented an interactive tool that assists users in drawing from photographs by providing guidance and corrective feedback inspired by traditional drawing techniques. Our system is based on automatically generated guidelines that apply to generic models. By combining pen-based interaction with a real-time registration algorithm, our tool encourages users to practice these techniques on model photographs of their choice.

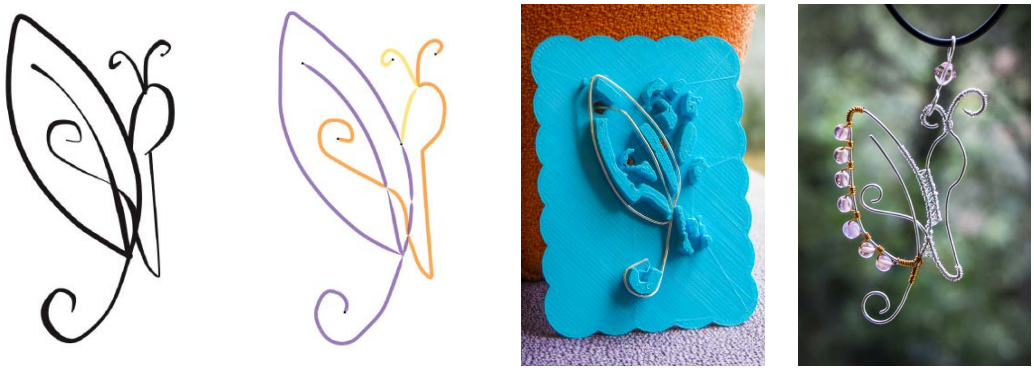
CHAPTER 3

Computer-Assisted Crafting of Wire Wrapped Jewelry

3.1 Introduction

Jewelry is one of the oldest and most prevailing forms of crafting, with the earliest known examples dating from almost 100,000 years ago. Traditionally, jewelry making has been a largely manual process that involves a range of techniques, such as mold-making, metal working, painting, etc. Today, while most commercial jewelry production leverages computer-aided design and manufacturing technology, there remains a large class of jewelry that continues to be made by hand. In recent years, interest in hand crafted jewelery has increased significantly with the growth of popular online crafting marketplaces such as Etsy and ArtFire, which allow independent artists to sell their work directly to consumers. Our goal in this work is to enable a broader range of users to create their own, customized handmade jewelry.

We focus on a specific style of jewelry making called *wire wrapping* that involves bending and connecting metal wires to create complex shapes (Figure 3.1d). Wire wrapping is one of the most popular forms of hand crafted jewelry; e.g., Etsy.com returns over 220,000 "wire wrapped jewelry" results. Moreover, since it involves affordable materials and does not require melting or soldering, wire wrapping is particularly appealing to casual crafters. Crafting sites like Instructables.com include hundreds of wire wrapping tutorials for creating a variety of jewelry and other ornaments. However, since most tutorials provide instructions for a given piece of jewelry and are hard to generalize, novices are limited to creating a fixed set of designs. In this Chapter, we present a computational design tool that empowers novices to create wire-wrapped jewelry from their own



(a) Input bitmap (b) Labeling (c) Printed jig (d) Fabricated piece

Figure 3.1: Our system allows novices to create a variety of custom jewelry. Following aesthetic and fabrication principles, our algorithm decomposes an input line drawing (a) into smooth, well-connected paths that cover each line in the drawing exactly once (b). We extrude support walls inside the sharp turns of the paths to create a physical *jig* that guides wire wrapping (c). Assembling the three wires of this design yields a butterfly pendant (e).

designs. Since wire wrapping can be viewed as a form of line drawing (where wire takes the place of ink or graphite), we allow users to specify their target designs as line drawings. There are two major challenges in designing and fabricating wire-wrapped jewelry from line drawings.

Wire decomposition. The first and most critical step is to create a *wire decomposition* of the drawing into the appropriate set of wires. Since pieces made of many wires are unstable and difficult to assemble, good decompositions usually consist of few wires. Yet, many shapes cannot be represented with a single wire without doubling back over parts of the drawing, which often detracts from the aesthetics of the resulting jewelry. Moreover, it is hard to bend a single piece of wire to create sharp angles. Effective decompositions require balancing these constraints and objectives.

Wire bending. Given a wire decomposition, the next step is to bend each piece of wire to match the shape of the path specified in the design. To help create smooth curves, jewelry makers often wrap wires around tools called *jigs*. Several companies like WigJig [WigJig 2015] sell generic boards and cylindrical pegs that support custom jig configurations. However, such pre-defined kits are not flexible

enough to create arbitrary shapes. For instance, cylinders can only constrain bends of constant curvature, and target shapes with multiple nearby bends can result in collisions between the pegs.

We propose an end-to-end design and fabrication system to help novices address the above challenges. The main contribution is an algorithm to automatically decompose an input drawing into wires. We formulate the decomposition task as a graph labeling problem, where the labels define groups of line segments, each of which should be "drawn" with a single wire (Figure 3.1b). Our aesthetic and fabrication objectives result in both soft and hard constraints, some of which have a global impact on the labeling. We describe a stochastic optimization that handles these constraints.

Another key feature of our system is the automatic generation of a custom 3D-printed jig from a given wire decomposition. Our jigs include a set of support walls that constrain the shape of the wire to match the design. We generate support walls strategically only on curvy portions of each wire path (Figure 3.1c) so that there is enough free space to easily manipulate the wire. Minimizing the amount of support geometry also reduces the physical material required to 3D print the jig. As extra guidance, our system generates instructions to help plan and execute the wire wrapping and assembly. Note that our approach differs from most recent work on computer-aided fabrication since our goal is not to 3D print the final jewelry piece, but rather to create an intermediate support structure that facilitates the hand wrapping process while preserving the joy of crafting.

We evaluated our approach with novice users with little or no training in jewelry design and crafting. Participants took several minutes to segment a design by hand. While they obtained similar solutions to ours on simple designs, our algorithm yields solutions with fewer wires or increased robustness on more complex examples. All participants were able to create a piece of jewelry in less than half an hour using a wire decomposition and custom jig generated with our system. Participants commented that our system provides clear instructions and that the jig is very helpful in creating the final wire-wrapped jewelry.

3.2 Related Work

Professional jewelry is a major industry for which dedicated CAD systems exist, such as GemVision Matrix and ArtCAM JewelSmith to name a few. These systems provide advanced features to model common types of jewelry (rings, bracelets, pendants, etc), to decorate shapes with relief and gems, as well as to render the design realistically.

Our work targets a different audience: novice crafters or hobbyists who wish to create their own unique jewelry with affordable materials and hands-on techniques. Studies on the *Do-It-Yourself* community suggest that manipulating materials by hand contributes to the pleasure and pride of crafting [Tanenbaum 2013].

Traditional crafting, such as wood working, sewing or metal smithing, require significant expertise both to manipulate physical materials and anticipate their behavior. Computer-aided design has the potential of making such crafting accessible to novices by simulating the end artifact as well as guiding its fabrication [Schmidt 2013]. A typical example is *Plushie* [Mori 2007], an interactive system to design plush toys that combines inflation simulation and geometric constraints to generate developable patches that reproduce a target shape after sewing. Skouras et al. [Skouras 2012; Skouras 2014] follow a similar workflow to assist the design of inflatable structures. Other recent fabrication-oriented design systems assist the creation of furniture [Umetani 2012], pop-up cards [Li 2011], paper airplanes [Umetani 2014], mechanical characters [Coros 2013]. Closer to our application domain, Beady [Igarashi 2012] assists the construction of customized 3D beadwork by decomposing a 3D model into strips of beads while simulating physical interactions between neighboring beads. We adopt a similar methodology to this family of work but apply it to the different domain of wire-wrapped jewelry. Given a line drawing, our tool automatically generates a decomposition into metal wires that satisfies artistic and fabrication constraints.

Computational tools have also been proposed to assist the assembly of physical objects. While early work focuses on the generation of instructions for existing models [Agrawala 2003], recent work integrates assembly constraints as part

of the design goals, for instance to create sculptures made of planar slices [Hildebrand 2012] and interlocking furniture [Fu 2015]. Craftsmen also often rely on intermediate support structures, or scaffolds, to assist assembly. Inspired by masonry techniques, Deuss et al. [Deuss 2014] rely on temporary chains to guaranty stability during the assembly sequence of self-supporting structures. Temporary wooden structures have also been used for the fabrication of wire mesh sculptures [Garg 2014]. Taking inspiration from traditional wire-wrapping techniques, our system automatically generates custom support structures, called *jigs*, to guide the bending of wires.

One of the objectives of our algorithm is to wrap the wire over each line of the input drawing exactly once. Similar objectives appear in the computational design of continuous line drawings [Wong 2011] and related travelling-salesman art [Kaplan 2005]. However, these algorithms aim to find a *single* path that traverses all edges or visit all vertices of a graph, at the cost of adding new lines to the drawing if needed. In contrast, our algorithm allows the use of a variable number of paths to cope with shapes that cannot be covered by a single path.

3.3 Wire-Wrapping Principles

As discussed previously, the main challenge in creating wire-wrapped jewelry is to convert an input design, often represented as a line drawing, into a wire decomposition that can be fabricated.

Books [McIntosh 2007; Dismore 2011; DeField 2015] and online tutorials [WigJig 2015; Instructables 2015] provide many examples and recommendations for creating such wire decompositions. We studied this literature and identified three key characteristics of good decompositions.

Low complexity. Wire-wrapped jewelry should be made with a small number of wires because it is hard to join multiple wires in a robust, stable way. In addition, part of the beauty of wire-wrapped jewelry comes from

the intricate loops created by a long wire following a complex path. Ideally, a piece of wire-wrapped jewelry should be made from a single wire. However, many input designs cannot be reproduced using a single wire, unless the wire doubles back over parts of the design, which artists tend to avoid for aeasthetic reasons. Good decompositions use the minimum number of wires such that each part of the design is traversed exactly once.

Smoothness. While jewelry wire is made to be malleable, the physical resistance of metal prevents the creation of sharp bends. The path of each wire in the decomposition should thus be as smooth as possible with sharp angles in the input design represented by wire crossings. In the inset, the wire follows the blue direction rather than the red one as it results in a smoother trajectory.

Robustness. A piece of jewelry is most robust when it is composed of a single wire. In cases where several wires are needed to create a shape, craftsmen try to connect each wire at least twice to other wires to avoid weak dangling segments. At each connection, thin wire is wrapped around the various wire segments to hold them together without soldering.

3.4 Wire Decomposition

3.4.1 Line-Drawing Vectorization

Our system takes as input a bitmap line drawing, which we assume to be made of a single connected component. We first convert this line drawing into a vector representation by applying morphological thinning [Soille 2003], chaining pixels between junctions and fitting Bezier curves on the resulting pixel chains. The output of this vectorization is an undirected graph where each vertex corresponds to a junction and each edge corresponds to a Bezier segment.

3.4.2 Energy Formulation

We denote the graph extracted from the input drawing as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ are the vertices and $\mathcal{E}$ the edges. Our goal is to decompose the graph into N sub-graphs, each sub-graph corresponding to a single wire of the final design. We express this problem as assigning a label $l_{n\in[0,N-1]}$ to each edge of the graph. We evaluate the quality of a given assignment $l \in \{l_{n\in[0,N-1]}\}^{card(\mathcal{E})}$ with three energy terms that correspond to the three design principles identified in Section 3.3. We now describe each of our energy terms and later provide details about the optimization procedure. Figure 3.2 illustrates the effect of each of the terms on the solution.

Low complexity. The first and foremost objective of our algorithm is to segment the graph into a small number of sub-graphs, such that within each sub-graph, a piece of wire can traverse every edge exactly once. In other words, the wire should form an *Eulerian path* through each sub-graph. The necessary condition for a graph to admit an Eulerian path is that it has either no odd degree vertices (for a closed path) or exactly two odd degree vertices that correspond to the two

Figure 3.2: Effect of each energy term. A trivial solution assigns a different label to each line segment (a). This line drawing cannot be represented by less than three Eulerian paths (b). In (b) the blue and purple paths contain sharp turns that would be hard to create with metal wire. Our smoothness term avoids sharp angles (c) but results in a dangling blue path for the antennas. Our robustness term encourages a solution where each path has at least two connections to other paths.

endpoints of an open path. This Eulerian path requirement makes our optimization particularly challenging as it introduces a complex, non-local hard constraint in our graph labeling problem. To address this challenge, we design our optimization procedure to only consider candidate decompositions where every sub-graph has an Eulerian path, as described in Section 3.4.3. Thus, our complexity term only needs to penalize solutions with a large number of labels

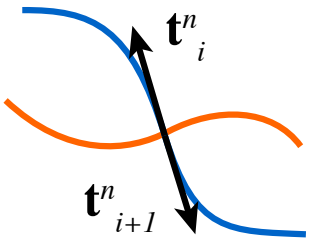
$$E_{complex}(l) = N. \tag{3.1}$$

Note that N is ultimately bounded by the number of edges in the graph, since the worst solution would be to assign a different label to each edge.

Smoothness. Our second term favors smooth paths by measuring the angle between consecutive curve segments at their junctions

$$E_{smooth}(l) = \sum_{n\in[0,N-1]} \sum_{i} \arccos \frac{\mathbf{t}_i^n \cdot \mathbf{t}_{i+1}^n}{|\mathbf{t}_i^n||\mathbf{t}_{i+1}^n|}, \tag{3.2}$$

where we loop over the Eulerian paths that correspond to the N labels and measure at each intersection the angle formed by the tangent $\mathbf{t}_i^n$ of the incoming curve segment and the tangent $\mathbf{t}_{i+1}^n$ of the subsequent outgoing curve segment. The inset illustrates our notation.



Robustness. We adopt a simple robustness heuristic which avoids dangling pieces by penalizing sub-graphs with less than two connections to other sub-graphs

$$E_{robust}(l) = \sum_{n\in[0,N-1]} \delta_n(l) \tag{3.3}$$

$$\delta_n(l) = \begin{cases} 1, & \text{if sub-graph } l_n \text{ has less than 2 connections} \\ 0, & \text{otherwise} \end{cases}$$

3.4.3 Optimization Method

Our optimization minimizes a weighted sum of the terms

$$\underset{l}{\operatorname{argmin}}\, E(l) = E_{complex}(l) + \lambda_s E_{smooth}(l) + \lambda_r E_{robust}(l) \qquad (3.4)$$
$$\text{s.t. each sub-graph } l_n \text{ admits a Eulerian path}$$

where λ_s and λ_r balance the contribution of the terms. The complexity and robustness terms have a global impact on the label configuration and as such prevent the use of standard graph-cut techniques. An exhaustive evaluation of the $N^{card(\mathcal{E})}$ configurations is also not feasible for any problem of reasonable size. Finally, as discussed previously, the Eulerian path requirement imposes a hard constraint on the labeling.

Algorithm 1 Optimization procedure

Generate initial configuration l (one path per edge)
Initialize solution $l_{min} = l$
Initialize relaxation parameter $T = T_{init}$
repeat
  Generate l' from l by applying a random operator (join, split or permute) on a random path
  Compute acceptance probability $R = \exp\left(\frac{E(l)-E(l')}{T}\right)$
  Draw a random value $p \in [0, 1]$
  if $p < R$ **then** update $l \leftarrow l'$
  else update $l \leftarrow l$
  Update relaxation parameter $T \leftarrow C \times T$
  if $E(l) < E(l_{min})$ **then** update $l_{min} \leftarrow l$
until $T < T_{end}$
return l_{min}

In light of these challenges, we adopt simulated annealing [Kirkpatrick 1983] to explore the space of configurations, as summarized in Algorithm 1. Our datastructure represents each sub-graph as an Eulerian path. To satisfy the Eulerian constraint, we initialize the optimization by assigning a different label to each edge of the graph, i.e. we have $N = card(\mathcal{E})$ sub-graphs of size 1, each being an Eulerian path by construction. At each iteration, the algorithm generates a new

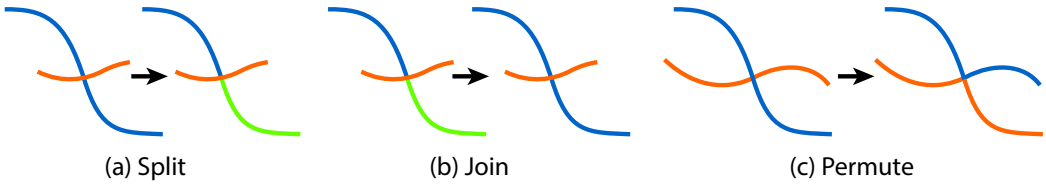


Figure 3.3: Our stochastic optimization relies on three operators to generate new configurations of labels. Colors represent different labels, i.e. different sub-graphs. Split creates two sub-graphs from a path (a). Join merges two paths in one (b). Permute exchanges half-paths between two sub-graphs (c).

configuration l' by perturbing the current configuration l. We carefully designed the perturbation operators to preserve the Eulerian property of the paths, as detailed in the next paragraph. The new configuration is always accepted if it decreases the energy. Configurations that increase the energy can also be accepted with a probability that depends on the energy variation between l and l' and a relaxation parameter T that geometrically decreases at a rate C. This acceptance strategy prevents the algorithm from getting trapped early in local minima. The algorithm returns the configuration with the minimum energy overall. We initialize T to 100 and C to 0.0001.

Perturbation operators. We define three local perturbation operators to generate new configurations: *join*, *split* and *permute*.

The *split* operator selects a random path and splits it into two paths at a random vertex (Figure 3.3a).

The *join* operator selects two random paths that share an end point and appends them to create a single path (Figure 3.3b).

The *permute* operator selects two random paths sharing a vertex and swaps the labels on either side of the vertex (Figure 3.3c). We randomly apply one of the three operators at each iteration. Note that a *join* can cancel the effect of a *split* and that applying *permute* twice on the same vertex in sequence has no effect, which is critical to allow the algorithm to escape bad configurations.

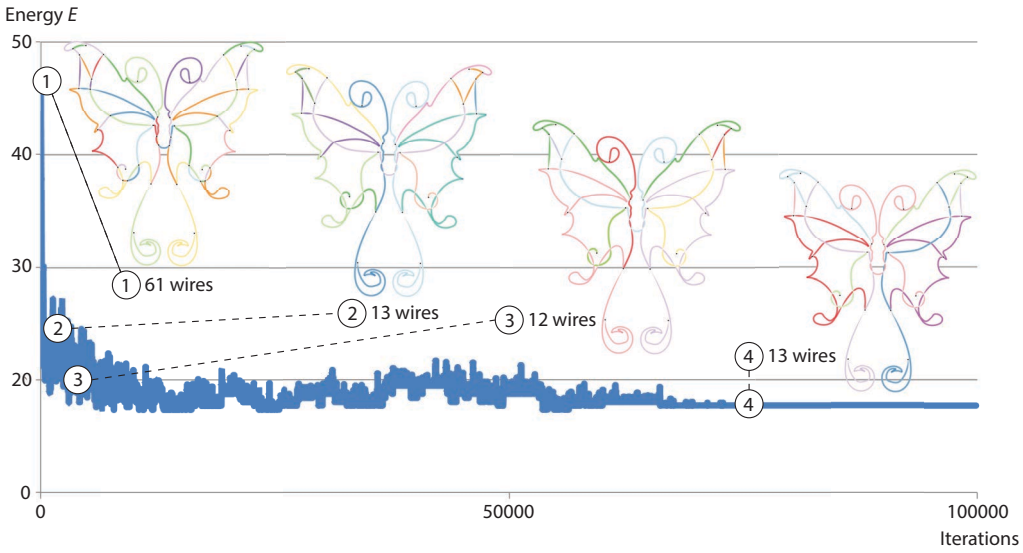


Figure 3.4: The energy decreases quickly during the first iterations as edges get connected into longer paths. The optimization then makes subtle updates to the configuration until it converges to a stable solution after 75000 iterations.

Since our perturbation operators never create new internal vertices of odd degree, our optimization is guaranteed to maintain the Eulerian property of the subgraphs.

Performance. Figure 3.4 shows the evolution of the configurations during the optimization. The energy makes large oscillations during the first iterations as the algorithm explores the solution space. It then makes more subtle adjustments and converges to a plateau as the relaxation parameter becomes selective. While our optimization is not guaranteed to reach the global minimum, in practice, it often finds solutions close to the optimum. Table 4.1 provides timings for representative drawings, ranging from 3 seconds for the smallest design to 32 seconds for the biggest one. Although the final number of wires in the decomposition depends on the complexity of the input drawing, our algorithm reduced the number of segments by four in these examples.

Drawing	Input segments	Output wires	Time (sec.)
Butterfly (Fig. 3.1)	12	3	2.7
Horse (Fig. 3.7)	24	5	12
Tiger (Fig. 3.5)	54	16	32

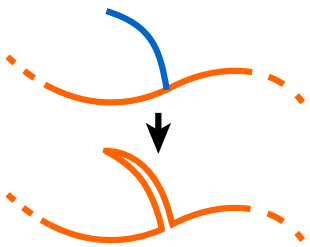
Table 3.1: Timing and number of input and output paths for a few representative input drawings.

3.4.4 Pre- and Post-Processing

We complement the algorithm described above with two optional features that expand the space of solutions and increase robustness.

Bridges. Complex shapes are sometimes impossible to represent with few Eulerian paths. Experienced jewelery makers often reduce the number of wires in the final design by adding short extra segments, or *bridges*, between nearby paths. To incorporate bridges, we pre-process the vectorized input drawing and detect pairs of vertices that are closer than $0.3\overline{\mathcal{E}}$, where $\overline{\mathcal{E}}$ is the average edge length. We only retain the pairs that are not connected by an edge and that are not separated by a line in the drawing. We insert these bridges as optional edges in the graph and augment our energy function with a term that penalizes their use. Denoting as $\mathcal{B}(l)$ the set of active bridges for a given label configuration, the penalty term is $E_{bridges}(l) = card(\mathcal{B}(l))$, weighted by $\lambda_b = 0.1$. We include the bridges in our optimization via two additional perturbation operators. The first operator enables a bridge, which allows subsequent iterations of the algorithm to join the bridge to another path. The second operator disables a bridge and creates two paths if the bridge was in the middle of a longer path.

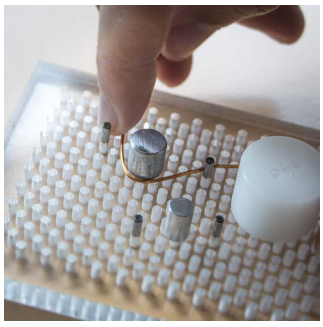
Junction refinement. While our robustness term favors well-connected sub-graphs, it is sometimes impossible to avoid dangling wires that are connected to one other sub-graph at a single point. Inspired by traditional practice, we strengthen such configurations as a post-process by merging the dangling segment with its connected segment and then doubling the wire over it, as illustrated in the inset. However, the double wire and sharp bends required by this solution often make the design less



aesthetically pleasing and harder to fabricate, which is why our optimization tries to avoid such cases.

3.5 Assisting Wire Bending

In addition to automating the decomposition of a design, our system also assists in its physical realization. The main challenge is in shaping the metal wire into the curves described by the design. A common technique to create smooth curves of a prescribed curvature is to wrap the wire around a curved tool, the so-called *jig*. We experimented with several alternatives to create jigs that best support the creation of a target path.



Pegs. Current practice for creating custom jigs involves fixing cylinders, called pegs, into a board [WigJig 2015]. Pegs of different radii constrain the wire to form loops of different curvature. The main advantage of this technique is that a generic set of pegs is sufficient to create a variety of shapes. However, this approach also imposes strong constraints on the design. First, cylindrical pegs can only produce turns of constant curvature. Second, a design with closely-spaced turns often results in colliding pegs. We originally considered moving pegs during fabrication to avoid collisions. However, optimizing peg placement and wrapping ordering is computationally complex. Our initial tests also revealed that the assembly sequence is too hard to follow while keeping the wire in place.

Naive extrusion. Given the increasing accessibility of 3D printing, an alternative solution is to generate and print jigs that are customized for a specific design. We first attempted to simply extrude the negative space around the design, which effectively produces a channel along the trajectory of the wire. Users thus need to *push* the wire into the channel, rather than *wrapping* the wire around pegs. However, our initial tests with this technique showed that pushing the wire creates jaggy curves because the wire isn't straightened by the longitudinal

tension of wrapping. A naive extrusion also requires printing support all along the wire, even in areas where no support is needed such as straight lines or the exterior side of a turn.

Extruding at curvature maxima. Based on the above experiments, our final solution is to extrude support material only on the interior side of each turn. We define turns as portions of curves for which the curvature exceeds a threshold, fixed to 0.01 in our implementation. We additionally create walls for segments longer than 2cm even if their curvature is below threshold. Unlike the peg board approach, this solution does not suffer from collisions and allows users to create curves of arbitrary shape. Moreover, the empty space around the extruded walls gives room to wrap the wire with tension to obtain a smooth result. We also add small holes to the jig to mark the starting points of each path and to hold the wire in place.

Finally, our interface integrates several convenient features to guide users during fabrication. For each path, we display the wrapping sequence step-by-step by highlighting successive curve segments. We also display the length of each wire, to which we add a few centimeters of margin to ease manipulation. We also show how to attach each wire to its neighbors.

3.6 Evaluation

We evaluate our approach in three ways: comparing our wire decompositions to ground truth designs from existing wire-wrapped jewelry; gathering feedback from novice users; and using our system to create several pieces of jewelry from input clipart drawings.

Decomposing existing designs. Our wire decomposition algorithm implements principles we deduced from the wire-wrapping literature and example designs. To assess the effectiveness of this algorithm, we created three ground truth decompositions by tracing line drawings over images of existing wire-wrapped jewelry,

creating one curve per wire of the jewelry (Figure 3.5). We then gave rasterized versions of the traced drawings as input to our system. Our results are near-identitical to the original decompositions.

User experience. We recruited five novice crafters to use our system and provide feedback on their experience. Three participants were female and two were male,

(a) Inspiration (b) Artist's decomposition (c) Our decomposition

Figure 3.5: We use existing jewelry (a) as inspiration to test our algorithm. . Our solution (c) closely matches the Artist's original decomposition (b). The decompositions are identical for the lion and the fish, while the left ear of the tiger results in a slightly different configuration than the original, although with the same number of wires.

aged between 21 and 33. Only participant P1 had any prior experience in jewelry making, while P1, P2 and P4 had some experience with soldering. None had done wire wrapping before.

To start, participants performed a warm-up fabrication task to gain familiarity with the challenges of bending and attaching wires. The task involved wrapping a butterfly design on an existing jig with one wire attachment in the center of the design (see inset). We provided participants with the jig, metal wire, pliers and a ruler. To facilitate the attachment step, we also provided a *third-hand*, an inexpensive crafting tool with two clips for holding different pieces of wire in place.

We then gave participants a line drawing and asked them to create a wire decomposition by hand according to our principles of few, smooth and well-connected wires. We provided a different input drawing for each participant to help us gain a broader range of feedback. The left two columns of Figure 3.6 compare our automatic decompositions with the user-generated results. Even for these relatively simple designs, our system produced different decompositions for three of the drawings. Our decompositions for the lotus (P3) and dolphin (P5) require fewer segments than the user segmentations. For the butterfly, P4 found a solution with the same number of wires as ours, but the purple antennae segment is not robust, as it only has one connection to the rest of the piece. We note that P3's two-wire solution for the lotus may have better preserved the rotational symmetry of the design after fabrication, which our algorithm doesn't model. We disabled bridges for this comparison since participants did not have the option of creating bridges.

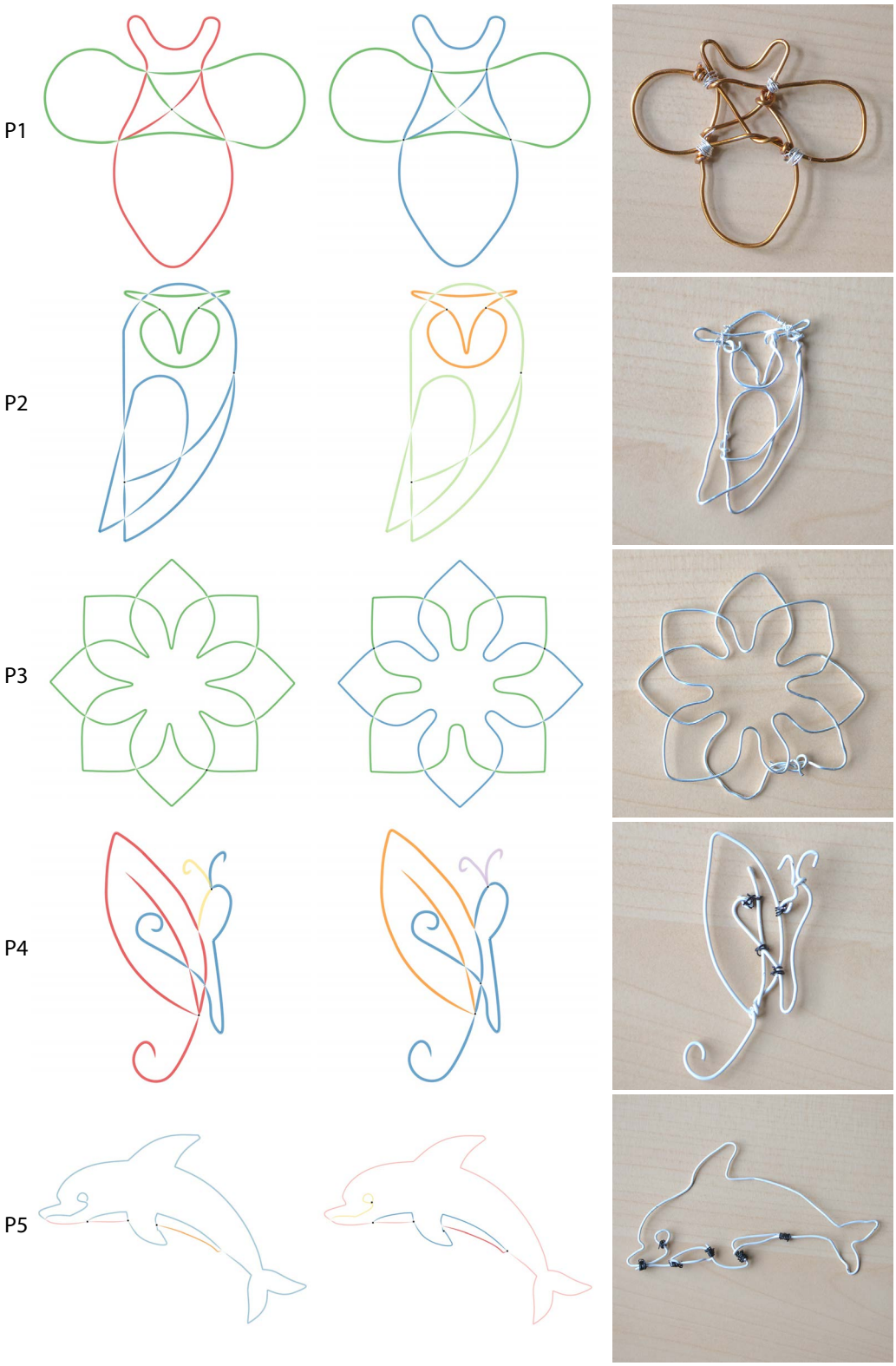


Figure 3.6: Results of our user experiment. For simple designs (P1, P2), participants found the same solution as our algorithm (a,b). For more complex designs, our algorithm finds a more robust solution (P4) or a solution with less wires (P5). All participants managed to create their jewelry in less than half an hour (c).

Finally, we asked participants to build the design from our decomposition and then conducted a post-study interview to gather feedback. Figure 3.6(c) shows the user-fabricated pieces of jewelry, which took about 20 minutes to create. Participants reported that the step-by-step visualization helped them understand the wrapping sequence of each path. On a Likert scale from 1 (strongly disagree) to 5 (strongly agree), three participants strongly agreed that the automatic decomposition helped in building the jewelry piece, one agreed and one had no opinion. P2 commented that *"My decomposition was the same as the automatic one, but the automatic solution gave me confidence in my choices and about where to end each piece."* Participants unanimously appreciated the jig, agreeing or strongly agreeing that it helps in building the jewelry piece. P2 commented that *"It would have taken much, much longer to build the jewelry without the jig"*. P1 noticed that *"The wire tends to jump out of the jig when there are too many layers of wire passing at the same point."*, which may be addressed by higher support walls.

Clipart to Jewelry. As a final means of evaluation, we used our system to convert clipart drawings downloaded from the Internet into wire-wrapped jewelry. We applied our vectorization algorithm on the bitmap images and manually cleaned-up little segments that would be hard to build at this scale. We also made small edits to some of the input drawings to ensure that they were made of a single connected component. Figure 3.1 and 3.7 show several complex jewelry pieces created with this approach by the first author.

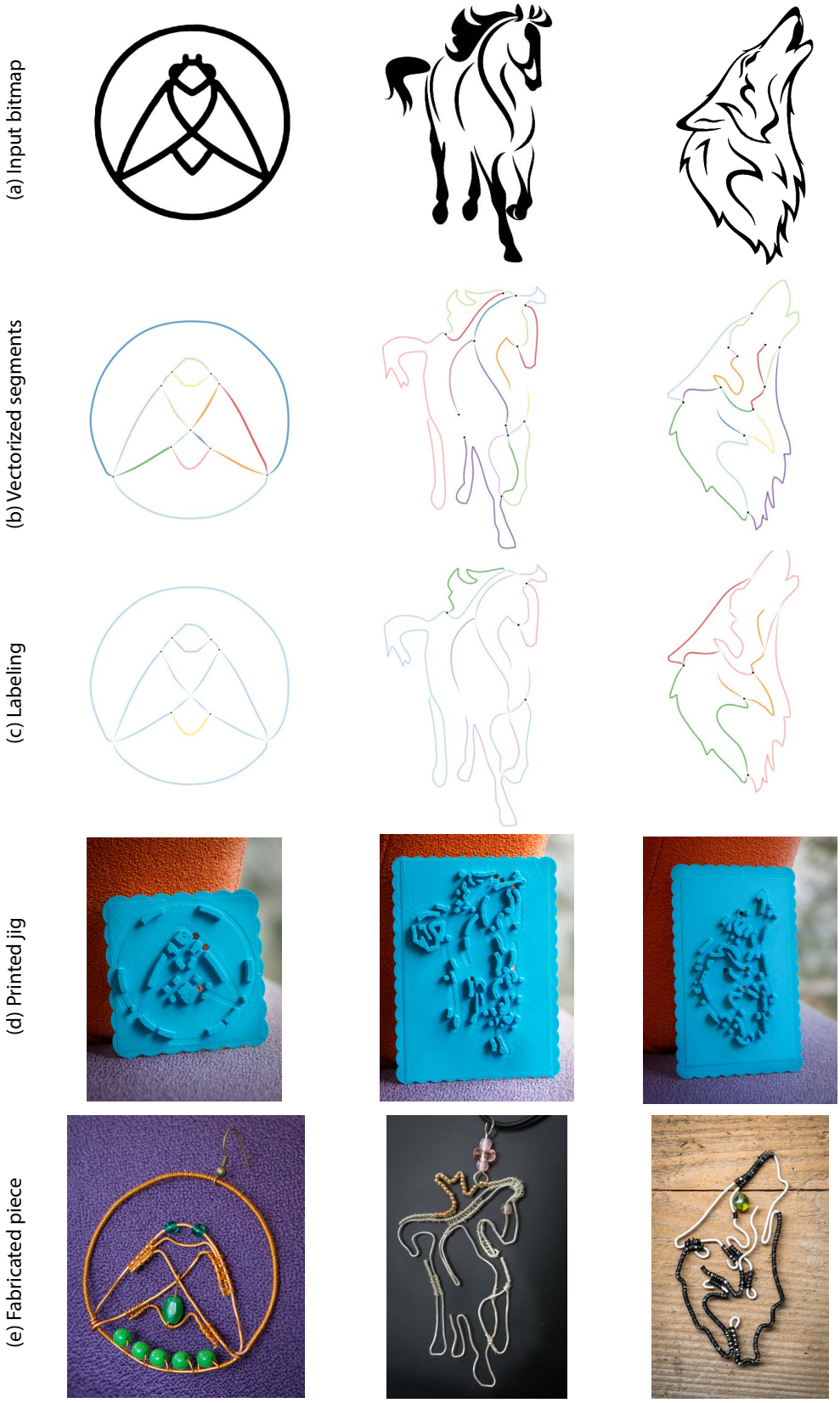


Figure 3.7: Clipart repositories provide a wide source of inputs for our approach. Note how our algorithm finds decompositions with few yet smooth and robust wires, such as the body of the bee and the legs of the horse.

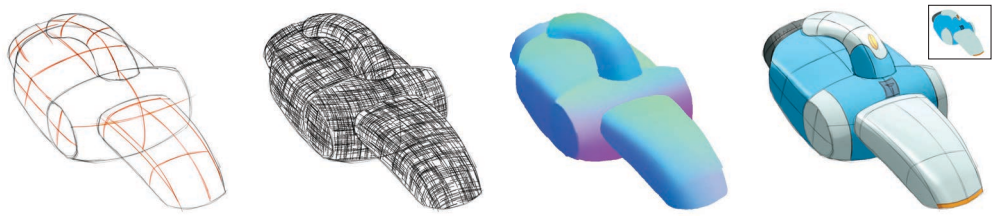
3.7 Conclusion

In this Chapter we presented an end-to-end system to assist the design and fabrication of wire-wrapped jewelry. From an algorithmic standpoint, our approach implements wire wrapping design principles in an optimization that segments an input drawing into a small number of wires. The optimization favors smooth wires that are well-connected to other wires to ease the fabrication of robust pieces. The optimization also strives to cover each line in the drawing exactly once to avoid double wires. From a fabrication standpoint, our system outputs a physical support structure that guides users in bending wires to obtain a desired shape. Our approach thus differs from completely automatic digital fabrication by preserving the hand manipulation of jewelry materials, which greatly contributes to the pleasure of craft.

CHAPTER 4

Regularized Curvature Fields from Rough Concept Sketches

4.1 Introduction



(a) Input sketch (b) Estimated curvature lines (c) Estimated normals (d) Shaded sketch

Figure 4.1: Designers commonly draw curvature lines to emphasize surface bending in concept sketches (a, red lines). We extrapolate these lines in a bitmap to form a dense cross field, from which we estimate the 3D curvature directions and the surface normal at each pixel (b,c). We use this information to compute shading, greatly enhancing the 3D look of the sketch (d). Note that this sketch is composed of several layers to represent the main body, handle and nozzle of the vacuum cleaner.

Designers frequently draw curvature lines to convey bending over smooth surfaces in concept sketches [Eissen 2011] (Figure 4.1(a)). We introduce a method to extrapolate strokes in a sketch to form a dense *cross field* that assigns two curvature lines to each pixel of the drawing while extending smoothly over flat and umbilical regions, where the lines of curvature are ill-defined (Figure 4.1(b)). By estimating this curvature information, our method enables the application of several 3D curvature-based algorithms to 2D drawings. For example, curvature lines have been used to guide parameterization [Ray 2006], texture synthesis [Lefebvre 2006], cross-hatching [Hertzmann 2000], and can

also provide surface normals for local shading. By applying these algorithms directly in the sketch, our approach allows designers to enhance the 3D look of their drawing during the early stages of design (Figure 4.1(d)), when 3D modeling is often distracting and time-consuming [Pipes 2007; Bae 2008; Shao 2012].

Our method takes as input rough bitmap sketches drawn on paper or with painting software, with lines often made of sketchy overlapping strokes. Our algorithm copes with such unorganized rough inputs in two steps. We first express the 2D projection of the two lines of curvature at each pixel as a scattered interpolation of the nearby lines. This interpolation results in a 2D *non-orthogonal* cross field, as the two projected curvature lines are not orthogonal due to foreshortening. We then lift this cross field to 3D by leveraging the fact that the lines of curvature should be orthogonal in 3D. We call the resulting 3D cross field over the image a *BendField*. We finally compute surface normals as the cross-product of the two 3D directions at each pixel.

While expressing our problem as a scattered interpolation makes our method robust to sketchy inputs, it requires us to address two challenges. First, each stroke in the sketch can only constrain one of the two lines of the cross field. Thus, for each pair of strokes there is a discrete choice of making them either parallel or transversal in the interpolation. Existing work in the context of *orthogonal* cross fields over 3D surfaces tackles this ambiguity either by nonlinear formulations that exploit orthogonality to map the lines into a space where they become equal [Ray 2009; Knöppel 2013] or by discrete variables that are part of the optimization [Bommes 2009]. We extend the second formulation to our context by introducing a new representation for non-orthogonal cross fields, using one angle to encode the orientation of an orthogonal cross and a second angle to encode the deviation from orthogonality.

The second challenge we address is the design of an interpolation energy that produces plausible lines of curvature at each pixel, subject to the constraints provided by the sketch. To do so, we conduct a mathematical analysis of curvature lines over smooth surfaces. We deduce the concept of *regularized curvature lines*

that model the way lines in a sketch align with curvature directions when these directions are well defined, and become geodesic, i.e. shortest path, in flat or umbilical regions. We then derive that, under parallel projection, the two families of regularized curvature lines that compose the 2D cross field are smooth along each-other. We use this specifically-designed measure of smoothness to extrapolate the lines over the sketch. The resulting cross field provides a vivid sense of the 3D shape, smoothly interpolating the prescribed curvature lines without introducing extraneous surface variations. We use our cross fields to enrich a variety of concept sketches with 3D-looking shading and texturing.

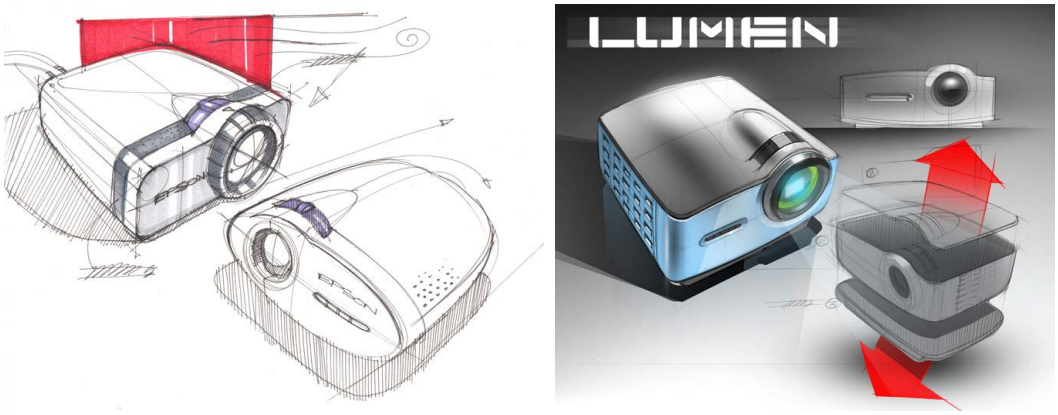
In summary, this Chapter introduces three contributions:

- A method to estimate 3D consistent curvature and normal fields from rough 2D concept sketches. In contrast to existing methods that require clean vectorial curves, our approach is able to handle sketchy drawings provided in bitmap form.
- A representation and optimization algorithm for non-orthogonal cross fields.
- The mathematical formulation of regularized curvature lines, from which we derive a novel smoothness energy to extrapolate curvature lines.

4.2 Related Work

Sketch editing. Designers commonly use line drawings to quickly explore 3D concepts without the burden of CAD modeling [Pipes 2007; Bae 2008]. While rough *ideation sketches* are often only made of lines (Figure 4.2(a)), shading and textures are subsequently painted to produce *presentation sketches* (Figure 4.2(b)) that better communicate 3D appearance to the clients and decision makers [Eissen 2011].

Several sketch-editing tools have been proposed to facilitate colorization, shading and texturing of line drawings. Scribble-based interfaces propagate colors in empty or uniformly-textured regions [Qu 2006; Sýkora 2009]. Inspired by modeling systems based on shape inflation [Igarashi 1999; Nealen 2007], *Lumo* and subsequent algorithms consider



(a) Rough ideation sketch (b) Shaded presentation sketch

Figure 4.2: Typical design sketches by Spencer Nugent on sketch-a-day.com©. (a) Designers draw rough ideation sketches to explore early concepts. (b) Shading is subsequently painted for presentation to decision makers.

that the lines in a drawing delineate an inflatable proxy on which they compute shading and texturing [Johnston 2002; Joshi 2008; Winnemöller 2009; Sýkora 2011].

Inflated normal maps have also been used to generate stylized shading that mimics artistic guidelines [Lopez-Moreno 2013] or that follows plausible shading flows [Vergne 2012]. Sýkora et al. [Sýkora 2014] further improve realism by generating a bas-relief proxy with consistent depth ordering that they use to compute global illumination effects. Finally, Cole et al. [Cole 2012] adopt an example-based approach to estimate normal fields from contour drawings of smooth abstract shapes. The above methods produce convincing results on cartoon blobby shapes solely defined by contours. In contrast, we target man-made shapes from concept sketches and leverage interior curvature lines to control the shape away from contours. To do so, we propose a novel smoothness energy that better preserves curvature lines than the harmonic energies used in shape inflation.

Closer to our work are the *CrossShade* and *True2Form* algorithms [Shao 2012; Xu 2014], which generate normal maps and 3D curve networks from concept sketches. Both methods work with vector drawings and estimate 3D information from intersecting curves locally aligned with curvature directions. Our method

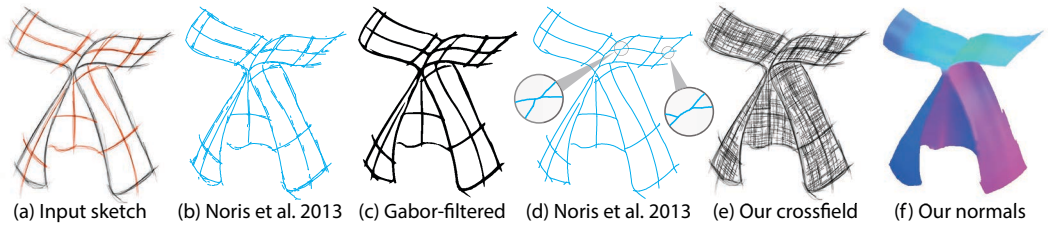


Figure 4.3: Limitations of vectorization. Rough sketches are made of many overlapping strokes (a) that vectorization algorithms [Noris 2013] interpret as multiple short curves (b). The Gabor filter of [Bartolo 2007] groups many strokes together but also tends to smooth junctions between intersecting strokes (c). As a result, junctions form ambiguous configurations (d, inset) that prevent the merging of segments into continuous curves suitable for CrossShade [Shao 2012]. We propose an alternative approach to directly estimate curvature information and normals from the rough sketch (e,f), alleviating the need for vectorization and manual cleanup. The results of [Noris 2013] were produced with default parameters.

targets a similar application domain as CrossShade but handles rough bitmap drawings rather than clean vector art. Designers often produce rough preliminary sketches in a bitmap form, either from scanned pen-on-paper drawings or from painting software, which requires less precision than vector tracing. Working with bitmaps requires us to formulate the extrapolation of curvature lines as a scattered data interpolation rather than the parametric Coons interpolation used by CrossShade.

While vectorization algorithms could be used to convert bitmaps into vectorial curves, state-of-the-art algorithms remain challenged by sketchy drawings [Noris 2013], or require the temporal information provided by digital sketching [Orbay 2011]. When applied on a rough sketch, the recent method by Noris et al. [Noris 2013] produces multiple curves in the presence of overlapping strokes (Figure 4.3(a,b)). While filtering the sketch can group overlapping strokes [Bartolo 2007], curve segments cannot be automatically merged at junctions because of ambiguous configurations (Figure 4.3(c,d)). As a result, vectorized segments should be manually edited and merged to form suitable input for CrossShade, which assumes that each curvature line is formed by a single curve. We designed our approach to bypass vectorization and avoid all these shortcomings

(Figure 4.3(e,f)).

Line fields in images. Kass and Witkin first proposed to analyze oriented patterns by computing a smooth *line field* (i.e. 2-direction vector field) perpendicular to the strong gradients in an image [Kass 1987]. Similar image-guided line and vector fields have been used for image filtering [Weickert 1999; Kang 2009; Kyprianidis 2011], edge detection [Kang 2007], painterly rendering [Haeberli 1990]. While we take inspiration from this body of work, our goal is to estimate a *cross field* rather than a line field, which requires us to assign the strong gradients in the image to one of two lines. The structure tensor [Harris 1988; Aach 2006] and streerable filters [Freeman 1991] can be used to estimate multiple orientations at corners and junctions but their response vanishes away from the image contours. In addition, notice that line fields are different mathematical objects than cross fields. Since line fields cannot represent the quarter-index singularities of cross fields, they are inappropriate in our context (cf. [Ray 2008]).

Line fields and cross fields on surfaces. Many surface-processing algorithms rely on the definition of smooth line fields or cross fields over 3D objects [Knöppel 2013]. These fields have been used to guide parameterizations [Ray 2006], texture synthesis [Praun 2000; Lefebvre 2006; Fisher 2007], cross-hatching strokes [Hertzmann 2000], quad remeshing [Alliez 2003; Bommes 2013b]. Most of these methods work with *orthogonal* cross fields aligned with the principal directions of curvature of a *known* 3D surface. Our goal is to allow the use of this family of algorithms directly in the 2D drawing of an *unknown* surface by estimating an orthogonal cross field from a sparse set of projected curvature lines. To achieve this goal, our algorithm first generates the *non-orthogonal* cross field which results from projection into the 2D drawing plane.

Liu et al. [Liu 2011] describe an algorithm to compute *conjugate* cross fields over 3D surfaces, which are non-orthogonal. Their approach generalizes the orthogonal cross field approach of Hertzmann and Zorin [Hertzmann 2000], which due to its nonlinear and non convex energy functional has the tendency to produce non-optimal additional singularities. We instead adopt a mixed-integer formula-

tion, which overcomes such problems using an iterative optimizer [Bommes 2009], where in each step a convex linear problem is solved.

Concurrently to our work, two alternatives to handle non-orthogonal cross fields were developed. Panozzo et al. [Panozzo 2014] introduce the concept of *frame fields*, which are non-orthogonal and non-unit-length generalization of cross fields. They model a frame field as a combination of an orthogonal cross field – generated with a mixed-integer algorithm [Bommes 2009] – and a harmonic deformation field that captures scaling and skewness. Instead of splitting the optimization into two subsequent parts, we directly generalize [Bommes 2009] to determine skewness and unit-length cross field in one combined step.

Diamanti et al. [Diamanti 2014] propose the powerful idea of *polyvector fields*, which encode arbitrary sets of vectors as roots of complex polynomials, and thus can handle non-orthogonal cross fields as a special case. This method also performs a single combined optimization but the harmonic interpolation of boundary constraints is done in the space of polynomial coefficients while we interpolate rotations in the more natural space of angles.

One major difference between the two concurrent approaches and ours is that both frame fields and polyvector fields are interpolated from sparse *frame constraints*, i.e. they require local constraints on both lines of the cross field. In contrast, our method needs to handle *partial* constraints since sketched strokes only constrain one of the directions of the field while leaving the transverse direction free for optimization. Another difference is that our angle-space parametrization enables the optimization of unit-length fields, which prevents undesired field shrinkage experienced by Diamanti et al. (see Figure 14 in their paper).

All the aforementioned methods use a harmonic energy to generate smooth fields. We show that applying a harmonic energy in our context does not produce plausible curvature lines as it does not account for the way such lines behave on a surface in 3D.

4.3 Overview

Designers extensively use free-hand sketches to quickly visualize the shape they envision [Eissen 2011]. Figure 4.2 shows a real-world example of a projector. In such sketches, skilled artists capture all surface information by drawing two types of lines:

- *Discontinuity lines* mark the sharp creases and silhouettes that delineate smooth surface patches (Figure 4.1a, black lines).
- *Curvature lines* convey bending within the surface patches and extend smoothly in flat or umbilical regions (Figure 4.1a, red lines).

While concept sketches are typically made of a sparse set of lines, they prove to be sufficient to describe 3D shapes since viewers mentally extrapolate the curvature lines to form a dense network on the imagined surface, assuming that the geometry of a curve is representative of the geometry of the surface in its vicinity [Stevens 1981; Bessmeltsev 2012]. Mathematically speaking, this network corresponds to a smooth *cross field*, which associates two orthogonal lines to each point of the surface. Our goal is to mimic viewers' inference to recover the cross field over the visible 3D surface conveyed by a concept sketch.

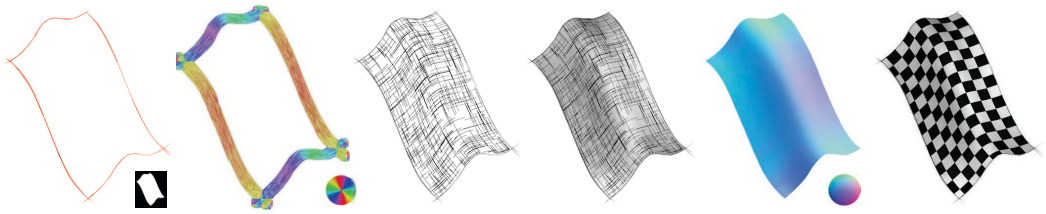
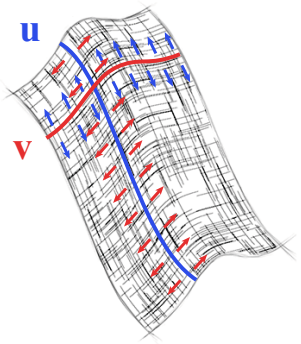


Figure 4.4: Our algorithm takes as input a bitmap sketch (a) from which we estimate the local orientation of the strokes (b). Assuming that the strokes represent curvature lines on an imaginary surface, our formulation extrapolates them as a non-orthogonal cross field that mimics the behavior of a projected curvature field (c). We then lift the cross field to 3D by imposing orthogonality constraints (d). We finally compute normals and texture parameterization from the 3D cross field (e,f).

Figure 4.4 illustrates the main steps of our algorithm. We take as input a bitmap line drawing, as commonly drawn on paper or with painting tools like Adobe Photoshop and Autodesk SketchBook, and a binary mask to identify pixels that belong to the object (Figure 4.4a). We additionally assume that users draw discontinuity and curvature lines in a different color and decompose complex models by drawing independent parts in different layers. Since designers draw in 2D, the curvature lines in the sketch only provide us with constraints on the *projected* cross field, which is non-orthogonal due to foreshortening. The first part of our algorithm consists in estimating this smooth non-orthogonal cross field from the local orientation of the lines in the sketch (Figure 4.4b,c). In a second step, we lift the cross field to 3D by assuming parallel projection and constraining the 3D lines to be orthogonal (Figure 4.4d). This 3D information allows us to apply several geometry-processing algorithms over the drawing, including the computation of surface normals for shading (Figure 4.4e) and parameterization for texture mapping (Figure 4.4f).

Our two main technical contributions from this Chapter are described in Sections 4.5 and 4.6. Section 4.5 introduces the concept of *regularized curvature lines* that encompass the curvature lines in a sketch and their extension as geodesics over flat or umbilical regions. From this concept we deduce a variational formulation for the non-orthogonal cross field. In a nutshell, our energy encourages the two families of lines that compose the cross field to be smooth along each other, as illustrated in the inset where the lines of the family $\mathbf{u}$ are smooth along the family $\mathbf{v}$ and vice-versa. We derive this property from the fact that curvature lines on a surface are free of geodesic torsion.



Section 4.6 then describes how to solve for the cross field that minimizes our energy subject to the stroke constraints. The main challenge we face is that, while each stroke constrains one of the two lines ($\mathbf{u}$,$\mathbf{v}$) in the cross field, we don't know which of the four directions $\{\mathbf{u}, -\mathbf{u}, \mathbf{v}, -\mathbf{v}\}$ is constrained a priori. We handle these discrete degrees of freedom using a mixed-integer formulation that jointly

solves for the smooth cross field and the assignment of each constraint to one direction. Our mixed-integer formulation is also necessary to handle singularities in the cross field, in which case the surface needs to be split into charts related by discrete permutations of the directions that form the cross field.

4.4 Stroke Constraints

Given a rough bitmap sketch, we first need to estimate the tangent of the strokes which will then act as constraints to align the cross field (Figure 4.4b). We obtain this information from the structure tensor, a popular tool to estimate local orientation in images [Kyprianidis 2011]. The structure tensor of an image $I(x, y)$ is expressed by means of the partial derivatives $I_x = \partial I/\partial x$ and $I_y = \partial I/\partial y$ as

$$S(I) = \begin{pmatrix} I_x \cdot I_x & I_x \cdot I_y \\ I_x \cdot I_y & I_y \cdot I_y \end{pmatrix}.$$

Its major and minor eigenvectors provide the directions of maximum and minimum change in the image. The tangent along a stroke is thus given by the minor eigenvector, except at corners and junctions where the presence of two orientations make the two eigenvalues almost equal. We use the magnitude of the minor eigenvalue to attenuate the influence of these unstable corners and junctions

(a) Input sketch (b) Local orientation (c) Constraint weight

Figure 4.5: We use the structure tensor to estimate the local orientation of the strokes in a sketch (b). We attenuate the strength of the orientation constraints near corners and junctions, where the estimation is unstable (c).

in the cross field estimation. In practice, we normalize the minor eigenvalue λ_2 at each pixel i by the maximum minor eigenvalue of the image to obtain a weight $w_i = 1 - \frac{\lambda_2(i)}{\max(\lambda_2)} \in [0, 1]$ that we apply on the orientation constraints.

We found that applying a bilateral filter on the structure tensor produces smoother estimates while preserving abrupt changes of direction. We used the same range parameter $\sigma_r = 2$ for all inputs, and different presets for the spatial extent σ_s to account for various levels of sketchiness, as discussed in Section 4.9. Figure 4.5 illustrates the local orientations and the attenuation weight at corners and junctions. Note that while we visualize the orientations as a wide strip around all strokes, we only apply constraints on the pixels covered by curvature strokes.

4.5 Estimating Curvature Fields

The 2D strokes in the sketch correspond to the projection of a subset of the curvature lines. The goal of this section is to derive a proper way of smoothly extrapolating the sparse strokes to the dense curvature network of the intended surface. This step, which results in a dense (non-orthogonal) cross field, is illustrated in Figure 4.4c. We first provide an intuitive motivation for our 2D smoothness energy and its relation to interpolants used in prior work (Section 4.5.1). We then provide a formal derivation of this energy from properties of curvature lines and fields (Section 4.5.2) and extend the energy to lift the cross field to 3D (Section 4.5.3).

4.5.1 Motivation for the BendField Energy

For clarity, we assume for now that each stroke constraint has been assigned to one of the two lines ($\mathbf{u}$,$\mathbf{v}$) of the cross field. Our goal is to generate a smooth field aligned with these constraints. Prior work on the design of line fields [Fisher 2007], cross fields [Knöppel 2013] and normal fields [Johnston 2002] use an harmonic energy to measure the smoothness of a field. In our context, the harmonic energy independently penalizes strong gradients in the $\mathbf{u}$ and $\mathbf{v}$ fields

$$E_h = \int ||\nabla \mathbf{u}||^2 + ||\nabla \mathbf{v}||^2.$$

Figure 4.6(a) illustrates the behavior of this energy that tends to flatten the surface away from the constraints. To prevent such flattening, Shao et al. [Shao 2012],

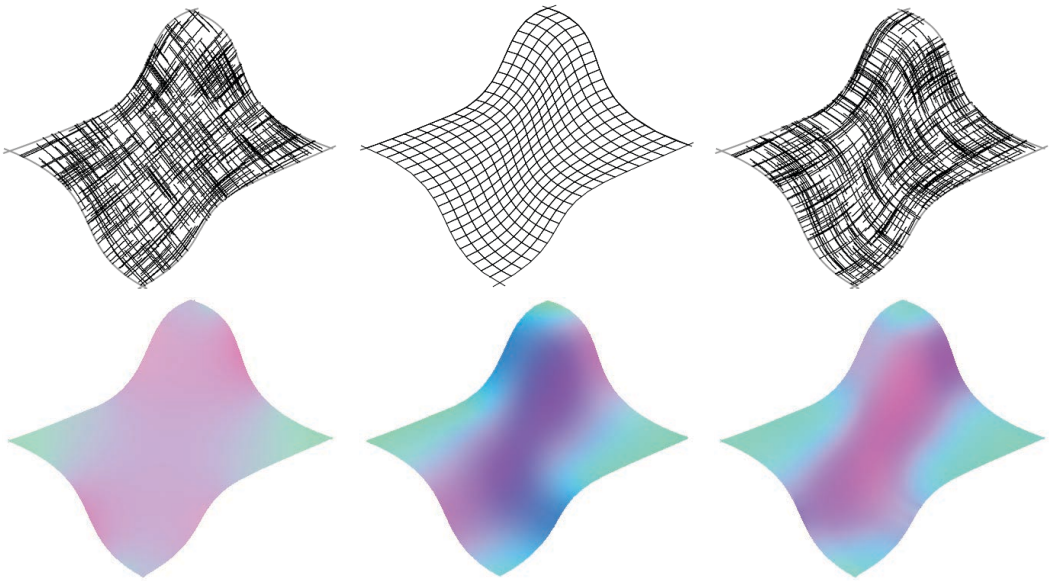


Figure 4.6: The harmonic energy produces a flat surface patch that does not preserve curvature away from the strokes (a). Prior work uses Coons patches to better capture the directionality of the boundary curves (b) [Shao 2012]. Our energy produces a similar interpolation by making the $\mathbf{u}$ and $\mathbf{v}$ vector fields smooth along each-other (c).

Biard et al. [Biard 2010] and Bessmeltsev et al. [Bessmeltsev 2012] interpolate normals and surfaces over quads bounded by curvature lines using parametric Coons patches [Farin 1999]. Since Coons patches interpolate the boundary segments linearly, they naturally align the ($\mathbf{u}$,$\mathbf{v}$) iso-lines to these boundaries, as shown in Figure 4.6(b). This alignment corresponds well to viewer expectation that a given curve is representative of other curves in its vicinity [Stevens 1981]. We designed our smoothness energy to produce a similar alignment in a scattered interpolation fashion. More precisely, our *BendField* energy relies on the *covariant derivatives* $\nabla_{\mathbf{u}}\mathbf{v}$ and $\nabla_{\mathbf{v}}\mathbf{u}$ to measures the smoothness of the vector field $\mathbf{u}$ along the streamlines of $\mathbf{v}$, and vice versa

$$E_{\text{bend2D}} = \int ||\nabla_{\mathbf{u}}\mathbf{v}||^2 + ||\nabla_{\mathbf{v}}\mathbf{u}||^2$$

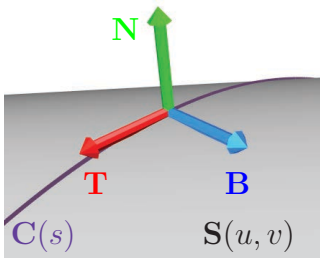
where $\mathbf{v} = (v_x, v_y)^T$ and

$$\nabla_{\mathbf{u}}\mathbf{v} = \begin{pmatrix} \partial v_x/\partial x & \partial v_x/\partial y \\ \partial v_y/\partial x & \partial v_y/\partial y \end{pmatrix} \mathbf{u}.$$

Note that the covariant derivatives couple the two vector fields, which harmonic and biharmonic energies cannot do. While additional work is needed to formalize the connection between our energy and Coons patches, Figure 4.6(c) shows that they behave similarly, even though our algorithm and CrossShade [Shao 2012] do not produce strictly identical normals on such a complex freeform surface patch (see Section 4.9 for additional comparisons).

4.5.2 Formal Derivation of the BendField Energy From Properties of Curvature Lines and Fields

Curvature lines. Given a parameterized surface $\mathbf{S}(u, v)$ embedded in $\mathbb{R}^3$, all curves on this surface can be described by univariate functions $\mathbf{C}(s) := \mathbf{S}(u(s), v(s))$. For simplicity in the following we assume an arc length parametrization such that $s \in [0, L]$ with L being the length of $\mathbf{C}$. The curvature properties of such a curve w.r.t. to its surface are characterized by the behavior of the so called Darboux frame. This orthonormal frame $(\mathbf{T}, \mathbf{B}, \mathbf{N}) \in \mathbb{R}^{3\times3}$ consists of the unit tangent $\mathbf{T} = \frac{d\mathbf{C}}{ds}$, the surface normal $\mathbf{N}$ and the tangent normal $\mathbf{B} = \mathbf{N} \times \mathbf{T}$. While traversing the curve, this orthonormal frame undergoes rotations. The rotational speed around the axes of the frame are known as *geodesic torsion* $\tau_g = \mathbf{N} \cdot \frac{d\mathbf{B}}{ds}$, *normal curvature* $\kappa_n = \mathbf{N} \cdot \frac{d\mathbf{T}}{ds}$ and *geodesic curvature* $\kappa_g = \mathbf{T} \cdot \frac{d\mathbf{B}}{ds}$ for rotations around $\mathbf{T}$, $\mathbf{B}$ and $\mathbf{N}$ respectively. Figure 4.7 illustrates the geometric intuition behind these notions.



Important in our context is the observation that curvature lines are characterized by vanishing geodesic torsion. More precisely, a curve is a curvature line if and only if $\tau_g = 0$ [do Carmo 1976; Biard 2010]. The name curvature line reflects the fact that such a curve is always tangent to one of the principal curvature directions of the surface. Intuitively, a surface curve has non-zero geodesic torsion if the surface bends most in

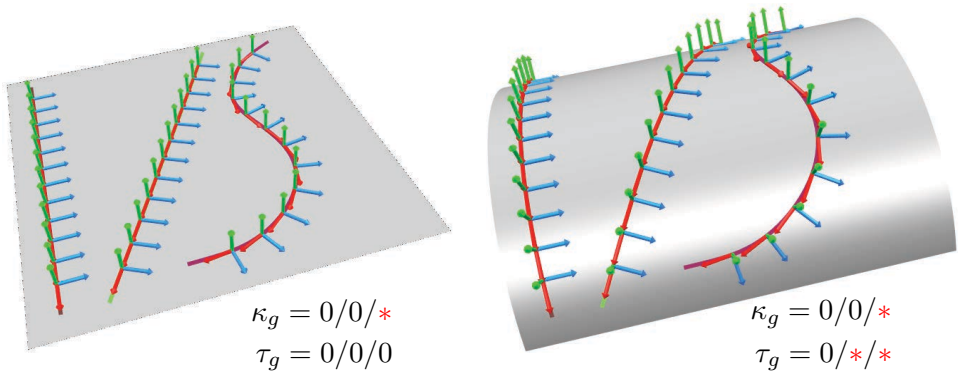


Figure 4.7: The geodesic curvature κ_g and geodesic torsion τ_g of three curves are listed from left to right with $*$ indicating nonzero. In the plane, only straight curves are geodesics ($\kappa_g = 0$) while every curve has $\tau_g = 0$ since the Darboux frame cannot rotate around T without changing the tangent plane. In (b) the plane is deformed to a cylinder. Now the second curve, although being geodesic, has a nonzero τ_g. Notice the implied rotation of B and N around T, which results from misalignment to the bending direction and accordingly vanishes for the curvature-aligned leftmost curve.

a direction that is not $\mathbf{T}$ nor $\mathbf{B}$ (Figure 4.7b), which contradicts the definition of curvature lines.

In each point of the surface where the two principal curvatures κ_1 and κ_2 are different ($\kappa_1 \neq \kappa_2$), exactly two unique curvature lines intersect. However, in flat and umbilical regions with $\kappa_1 = \kappa_2$ we have the ambiguity that every curve through these regions is a curvature line. This is also reflected by the fact that the geodesic torsion can be computed as $\tau_g = 0.5(\kappa_2 - \kappa_1)sin(2\theta)$ where θ is the angle between $\mathbf{T}$ and the direction of minimal curvature $\mathbf{K}_1$. Note that τ_g is proportional to the *curvature anisotropy* $\kappa_2 - \kappa_1$ and thus linearly vanishes in isotropically curved regions, independently of the tangent direction. We conjecture that designers avoid such ambiguity of curvature lines by sketching what we call *regularized curvature lines*, which we define more precisely next.

Regularized curvature lines. In areas of high curvature anisotropy, i.e. where $|\kappa_2 - \kappa_1|$ is large, designers sketch smooth curves that strictly follow principal curvature directions. However, the more isotropic the curvature gets, the more geodesic the sketched curves tend to be. Geodesic curves are characterized by

vanishing geodesic curvature $\kappa_g = 0$, i.e. the curve does not bend within the tangent plane. Therefore sketch curves behave like minimizers of the functional

$$E_\alpha = \int_C \tau_g^2 + \alpha \kappa_g^2 ds \tag{4.1}$$

The geodesic curvature κ_g behaves like a regularizer for τ_g since it becomes dominant in regions of isotropic curvature $\kappa_1 \approx \kappa_2$, where τ_g vanishes. We refer to this family of curves as *regularized curvature lines*, where α controls the strength of the regularization.

We hypothesize that designers apply a similar regularization when sketching because the trajectory of curvature lines is hard to predict in near-isotropic regions. This regularization toward geodesics is also supported by prior observations that designers draw curves aligned with curvature and geodesic lines [Shao 2012]. Similarly, perceptual studies suggest that people interpret intersecting curves in a drawing as aligned with principal directions of curvature [Stevens 1981; Mamassian 1998] and geodesics [Knill 1992]. This concept also provides a mathematical definition to the notion of *flowlines* mentioned in prior work [Bessmeltsev 2012; Zhuang 2013]. Finally, notice the closely related approach of modern quad meshing algorithms [Bommes 2013b] that align the quad mesh solely in anisotropic curvature regions while preferring smoothness everywhere else.

Regularized curvature cross field. Ultimately we are searching for the full (regularized) curvature network that extends the sketch curves to a dense orthogonal cross field. Therefore we have to extend the previous concept from curves on a surface $\mathbf{S}$ to cross fields that are tangent to $\mathbf{S}$. As an intermediate step first observe that the generalization to a unit-length tangent vector field $\mathbf{T}$, which is now defined for each point on S, yields the functional

$$E_\alpha(\mathbf{T}) = \int_{\mathbf{S}} \tau_g^2 + \alpha \kappa_g^2 dA = \int_{\mathbf{S}} (\mathbf{N} \cdot \nabla_{\mathbf{T}} \mathbf{B})^2 + \alpha (\mathbf{T} \cdot \nabla_{\mathbf{T}} \mathbf{B})^2 dA$$

where $\nabla_{\mathbf{T}}$ is the derivative along the streamline tangent to $\mathbf{T}$. This (extrinsic) directional derivative is necessary since the curves are now only implicitly defined as the streamlines of $\mathbf{T}$. Since a cross field is nothing more than a vector field on a 4-sheeted covering [Kälberer 2007] that can locally be parameterized by two vector

fields $\mathbf{U}$ and $\mathbf{V}$, we end up with

$$E_\alpha(\mathbf{U},\mathbf{V}) = \int_{\mathbf{S}} (\mathbf{N}\cdot\nabla_{\mathbf{U}}\mathbf{V})^2 + (\mathbf{N}\cdot\nabla_{\mathbf{V}}\mathbf{U})^2 + \alpha((\mathbf{U}\cdot\nabla_{\mathbf{U}}\mathbf{V})^2 + (\mathbf{V}\cdot\nabla_{\mathbf{V}}\mathbf{U})^2)dA. \tag{4.2}$$

Notice that in case of cross field singularities it is not possible to globally represent a smooth cross field by two smooth representative vector fields $\mathbf{U}$ and $\mathbf{V}$. This technical problem, however, can be easily handled by splitting the surface into coordinate charts that are related by discrete *transition functions* that permute the vectors of the cross field to align a cross in one chart to a cross in another chart [Ray 2006; Bommes 2009]. We explain the concept of transition functions in detail in Section 4.6.

Difficulty of sketch reconstruction. Equipped with the mathematical description of the 3D curvature network that is intended by the given 2D strokes we are now ready to state the optimization problem of sketch reconstruction. We are searching for a minimizer of $E_\alpha(\mathbf{U},\mathbf{V})$ subject to local unit length and orthogonality constraints $||\mathbf{U}|| = ||\mathbf{V}|| = 1$, $\mathbf{U}\cdot\mathbf{V} = 0$ and $\mathbf{N} = \mathbf{U}\times\mathbf{V}$, where we additionally require that the 2D strokes locally align to the 2D projection of $\mathbf{U}$ or $\mathbf{V}$. This is a very hard nonlinear mixed-integer problem since both the surface S and its tangent cross field $(\mathbf{U},\mathbf{V}) \in \mathbb{R}^{3\times 2}$, including discrete transition functions between charts, are unknown. Instead of optimizing it directly, we aim at first optimizing for its 2D projection which can then be used to estimate an appropriate initial solution for the 3D problem.

2D projection of curvature cross field. By parallel projection $P((x,y,z)^T) = (x,y)^T$, the unknown surface $\mathbf{S}$ becomes a known part of the Euclidean plane. However, this simplification comes at the cost of a distorted metric. Due to foreshortening the projection heavily affects dot and cross products such that $E_\alpha(\mathbf{U},\mathbf{V})$ is useless for our 2D setting. However, by restricting to the case $\alpha = 1$ it is possible to obtain a suitable formulation with a stable behavior under projection.

First notice that for curves we have $\int_C ||\frac{d\mathbf{B}}{ds}||^2 ds = E_{\alpha=1}$. This can be verified by projecting $\frac{d\mathbf{B}}{ds}$ on the orthonormal basis $\mathbf{T},\mathbf{B},\mathbf{N}$ and exploiting $\frac{d\mathbf{B}}{ds}\cdot\mathbf{B} = 0$ since

$\mathbf{B}$ is unit length. This means that for $\alpha = 1$, Equation (4.1) becomes

$$E_1 = \int_C ||\frac{d\mathbf{B}}{ds}||^2 ds = \int_C ||\nabla_{\mathbf{T}}\mathbf{B}||^2 ds$$

or equivalently for the cross field case

$$E_1(\mathbf{U}, \mathbf{V}) = \int_{\mathbf{S}} ||\nabla_{\mathbf{U}}\mathbf{V}||^2 + ||\nabla_{\mathbf{V}}\mathbf{U}||^2 dA.$$

Intuitively, our BendField energy favors parallelism of one vector field along the streamlines of the other one, which is consistent with the fact that non-parallelism can only be introduced by a non-zero geodesic torsion or geodesic curvature of the streamlines. Since parallelism is preserved by parallel projection, we can measure exactly the same quantity in a 2D projection, leading to

$$\boxed{E_{\text{bend2D}}(\mathbf{u}, \mathbf{v}) = \int_I ||\nabla_{\mathbf{u}}\mathbf{v}||^2 + ||\nabla_{\mathbf{v}}\mathbf{u}||^2 dA} \tag{4.3}$$

where I is the image plane, $\mathbf{u} = P(\mathbf{U})$ and $\mathbf{v} = P(\mathbf{V})$ are 2D projections of $\mathbf{U}$ and $\mathbf{V}$, $\nabla_{\mathbf{u}}\mathbf{v}$ and $\nabla_{\mathbf{v}}\mathbf{u}$ are covariant derivatives

$$\nabla_{\mathbf{u}}\mathbf{v} = \begin{pmatrix} \partial v_x/\partial x & \partial v_x/\partial y \\ \partial v_y/\partial x & \partial v_y/\partial y \end{pmatrix} \mathbf{u}$$

where $\mathbf{v} = (v_x, v_y)^T$. The optimization of Equation (4.3) is based on our novel non-orthogonal cross field representation, which is the topic of Section 4.6. Since lengths and angles are not preserved by parallel projection, the unit-length and orthogonality constraints can be omitted for the 2D optimization. However, they will be reinjected in the following step, which lifts the 2D minimizer of Equation (4.3) to 3D. Note also that we ultimately minimize E_{bend2D} subject to the stroke constraints, which prevents the trivial solution of a null cross-field.

4.5.3 Lifting the Cross Field to 3D.

Solving the previous optimization problem provides us with a good estimate of the 2D projection of $\mathbf{U}$ and $\mathbf{V}$. We obtain a local 3D estimate based on the knowledge that, in 3D, $\mathbf{U} \cdot \mathbf{V} = u_x v_x + u_y v_y + u_z v_z = 0$ due to orthogonality, and the additional assumption that designers favor viewpoints that minimize overall foreshortening [Eissen 2011; Shao 2012]. The minimal foreshortening tells us that $u_z = -v_z$ if the $\mathbf{u}$

and $\mathbf{v}$ vectors form an angle of less than $\frac{\pi}{2}$, while we have $u_z = v_z$ otherwise. Combining both constraints leads to a quadratic equation with two potential solutions $u_z = \pm\sqrt{|\mathbf{u} \cdot \mathbf{v}|}$. These two solutions reflect the global ambiguity between a convex and a concave surface patch, which cannot be resolved from the sketch alone [Shao 2012]. We obtain a globally-consistent solution by selecting for each pixel the candidate that produces the smoothest u_z field overall, subject to a few user indications to distinguish between the convex and concave interpretation (Section 4.8). We express this problem as a binary labeling, which we solve with [Kolmogorov 2006] as described in the Appendix.

Finally we use the globally consistent and normalized estimates $\mathbf{U} = (u_x, u_y, u_z)^T$ and $\mathbf{V} = (v_x, v_y, v_z)^T$ as an initial solution for optimizing energy (4.2). While in theory we should constrain $\mathbf{U}$ and $\mathbf{V}$ to have unit length, we found that the optimization can be greatly simplified by ignoring this constraint and by solely optimizing for u_z and v_z while keeping the 2D components constant. As a positive side effect, this choice not only regularizes the resulting cross field in length but also in direction. Therefore in addition we can safely simplify the functional by dropping the non-linear geodesic terms belonging to α. We still don't know the surface $\mathbf{S}$ such that we again approximate the energy over the image domain I, leading to

$$\boxed{E_{\text{bend3D}}(u_z, v_z) = \int_I (\mathbf{N} \cdot \nabla_{\mathbf{u}} \mathbf{V})^2 + (\mathbf{N} \cdot \nabla_{\mathbf{v}} \mathbf{U})^2 + \varepsilon_f (u_z^2 + v_z^2) dA} \tag{4.4}$$

where the last term is weighted by $\varepsilon_f = 0.005$ to weakly regularize the solution towards minimal foreshortening, as in [Shao 2012]. This is a hybrid formulation, where we compute a 3D cross field over the 2D image domain. Accordingly, the covariant derivatives are given by

$$\nabla_{\mathbf{u}} \mathbf{V} = \begin{pmatrix} \partial v_x/\partial x & \partial v_x/\partial y \\ \partial v_y/\partial x & \partial v_y/\partial y \\ \partial v_z/\partial x & \partial v_z/\partial y \end{pmatrix} \mathbf{u}$$

with $\mathbf{V} = (v_x, v_y, v_z)^T$. Note that thanks to our simplifications this energy is quadratic in the unknown z components, which can be verified by considering that $\mathbf{N} = \mathbf{U} \times \mathbf{V}$ and that $\mathbf{u}$ and $\mathbf{v}$ are constant. We optimize this functional subject

to nonlinear orthogonality constraints $\mathbf{U} \cdot \mathbf{V} = 0$ by an interior point method as described in more detail in Section 4.8. We finally normalize the resulting $\mathbf{U}$ and $\mathbf{V}$ vectors and compute the surface normal from their cross product.

4.5.4 Algorithm Overview

The ultimate goal of our algorithm is to extrapolate the given 2D strokes into a 3D BendField. Since 3D BendFields behave strongly nonlinearly it is necessary to split the overall task into smaller steps that enable mathematical formulations where poor local minima can be avoided. The idea is to start with convex approximations that reliably lead to good initial solutions for the subsequent nonlinear steps. The following overview, which gives a preview to Section 4.6, clarifies how we split the optimization:

BendField Algorithm

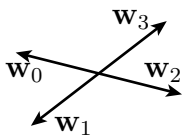
1. Estimate stroke constraints - Section 4.4
2. Optimize 2D BendField - Eqn. (4.3)
   (i) optimize unit-length harmonic cross field - Eqn. (4.5)
   (ii) refine to free-length 2D BendField - Eqn. (4.6)
3. Optimize 3D BendField - Eqn. (4.4)
   (i) local 3D estimate - Section 4.5.3
   (ii) refine to 3D BendField - Eqn. (4.4)
   (iii) compute normal field from $\mathbf{N} = \mathbf{U} \times \mathbf{V}$

4.6 Non-Orthogonal 2D Cross Fields

We now describe our representation of non-orthogonal 2D cross fields and their optimization subject to the user constraints. The goal of this step is to find a free-length non-orthogonal 2D cross field that minimizes the complicated nonlinear energy (4.3) while avoiding local minima. Therefore, we first generate a suitable initial guess by solving for a *unit-length* non-orthogonal cross field that minimizes a harmonic smoothness energy, which is convex up to integer

variables. We adopt the established greedy strategy of [Bommes 2009] to solve for the integer unknowns, which are then kept fixed during the subsequent nonlinear optimization. Notice that only the non-linear optimization is specifically tailored for our application, while the rest is a novel generalization of [Bommes 2009] to non-orthogonal cross fields and as such useful for many other applications.

Vector representation. A unit-length non-orthogonal cross corresponds to four unit-length vectors $\mathbf{w}_0$, $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ with the anti-symmetry conditions $\mathbf{w}_0 = -\mathbf{w}_2$ and $\mathbf{w}_1 = -\mathbf{w}_3$. Due to this anti-symmetry, an ordered tuple $[\mathbf{w}_0, \mathbf{w}_1] \in \mathbb{R}^{2\times 2}$ is locally sufficient to uniquely represent a non-orthogonal cross.



However, in the presence of singularities a smooth cross field cannot be globally represented by two smooth vector fields as illustrated in Figure 4.8(a). In addition, the lines in the sketch are unoriented, which prevents us from knowing which of the four vectors they should locally constrain (Figure 4.8(b)).

In order to handle such cases, we split the surface into charts which are connected by integer *transition functions* $T_{i\to j}$ [Ray 2006;

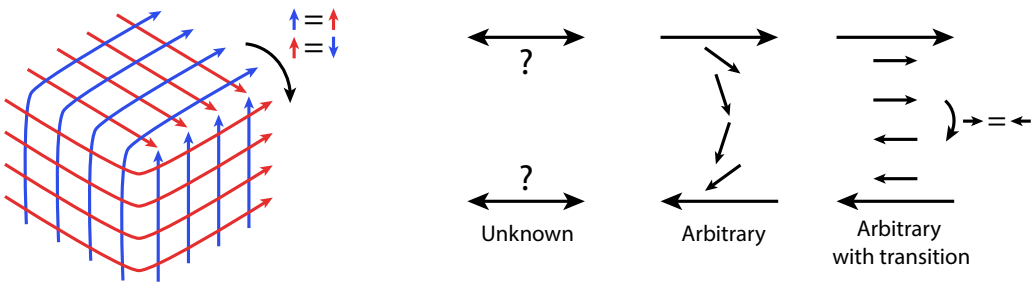


Figure 4.8: In the presence of a singularity, a smooth cross field cannot be globally represented by two smooth vector fields (a). The transition function permutes the vectors to best align them. Similarly, assigning the constraints to an arbitrary direction can produce unnecessary variation in a smooth vector field (b), which can be prevented by optimizing the transition functions. Note that while we illustrate the constraint assignment on a vector field, the same principle extends to cross fields where each constraint needs to be assigned to one of four directions.

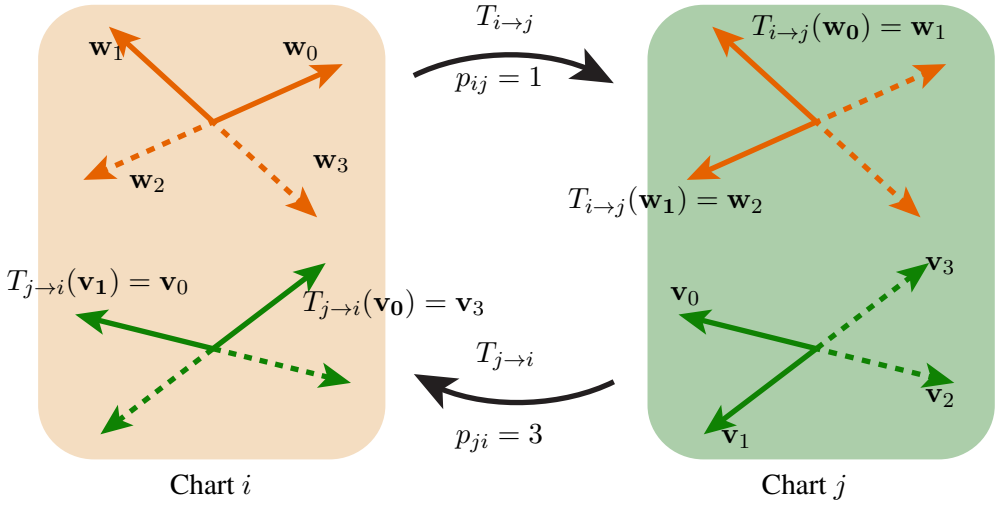


Figure 4.9: The transition function permutes the vectors of a cross in one chart to align it with a cross in another chart. The integer variable p_{ij} encodes the number of permutations to align a cross from chart i to chart j.

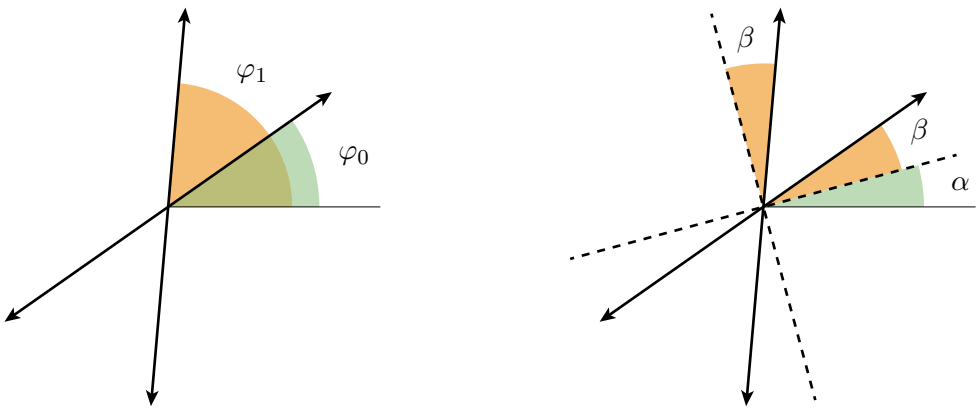
Bommes 2009], which cyclically permute the four vector fields p_{ij} times when moving from chart i to chart j, as illustrated in Figure 4.9. Formally this means $T_{i\to j}([\mathbf{w}_0, \mathbf{w}_1]) = [\mathbf{w}_{p_{ij}}, \mathbf{w}_{p_{ij}+1}]$ with $p_{ij} \in \mathbb{Z}$ and $\mathbf{w}_i = \mathbf{w}_{i+4}$ for all $i \in \mathbb{Z}$.

If we want to measure the similarity of two crosses $[\mathbf{w}_0, \mathbf{w}_1]$ and $[\mathbf{v}_0, \mathbf{v}_1]$ that are expressed w.r.t. charts i and j respectively, we need to consider the corresponding transition function. A fixed transition function enables the following convex similarity measure based on the Frobenius norm of matrices

$$||T_{i\to j}([\mathbf{w}_0, \mathbf{w}_1]) - [\mathbf{v}_0, \mathbf{v}_1]||_2^2 = ||[\mathbf{w}_{p_{ij}} - \mathbf{v}_0, \mathbf{w}_{p_{ij+1}} - \mathbf{v}_1]||_2^2$$

Angle representation. For unit-length cross fields it is often preferable to optimize in the space of polar coordinates (r, φ), where the unit-length constraint is simply $r = 1$. Consequently, a cross can be uniquely represented by two angles $[\varphi_0, \varphi_1]$. However in case of transition functions it is advantageous to choose a different parametrization of the two angles. We express a cross by the tuple $[\alpha, \beta]$, which is related to φ by

$$\varphi_i = \alpha + (-1)^i \beta + i \cdot \frac{\pi}{2}$$



(a) Polar representation (a) Our representation

Figure 4.10: A non-orthogonal cross can be represented by two angles $[\varphi_0, \varphi_1]$ in polar coordinates (a). To simplify formulas, we instead use one angle α to encode the global orientation of the cross (dashed lines) and one angle β to encode the deviation from orthogonality (b).

As illustrated in Figure 4.10, β describes the deviation from orthogonality while α can be understood as the closest orthogonal cross. One nice property is that for $\beta = 0$, our representation is exactly the same as the one used in [Ray 2008; Bommes 2009].

Similarly to the vector case we can define a smoothness measure between crosses $[\alpha_i, \beta_i]$ and $[\alpha_j, \beta_j]$ in different charts

$$\begin{aligned} E^{ij}_{\text{smooth}} &= ||T_{i\to j}[\varphi^i_0, \varphi^i_1] - [\varphi^j_0, \varphi^j_1]||^2 \\ &= ||[\varphi^i_{p_{ij}}, \varphi^i_{p_{ij}+1}] - [\varphi^j_0, \varphi^j_1]||^2 \\ &= (\alpha_i + (-1)^{p_{ij}}\beta_i + p_{ij}\frac{\pi}{2} - \alpha_j - \beta_j)^2 \\ &\quad +(\alpha_i - (-1)^{p_{ij}}\beta_i + p_{ij}\frac{\pi}{2} - \alpha_j + \beta_j)^2 \\ &= 2\left[(\alpha_i + p_{ij}\frac{\pi}{2} - \alpha_j)^2 + ((-1)^{p_{ij}}\beta_i - \beta_j)^2\right] \end{aligned}$$

Unit-length non-orthogonal cross fields in Images. In the image grid we assign one cross $[\alpha_i, \beta_i]$ per pixel p_i and assume that it is expressed w.r.t. its own

chart C_i. We obtain a finite difference approximation of the harmonic energy

$$E_h = \int_I ||\nabla\varphi_0||^2 + ||\nabla\varphi_1||^2 dA$$

on the regular image grid by summing the smoothness measure over all pixels i

$$E_{\text{smooth}} = \sum_i \sum_{j \in \mathcal{N}_i} E^{ij}_{\text{smooth}}$$

with $\mathcal{N}_i$ containing the upper and right neighbors of pixel i. Initially all transition functions are unknown. Thus, for an image with $n = w \times h$ pixels, we end up with an optimization problem of $2n$ continuous variables ($\alpha, \beta \in \mathbb{R}$) and $2n - w - h - 4$ discrete variables ($p_{ij} \in \mathbb{Z}$).

Furthermore, for a subset of pixels $S_c \subset I$ we have stroke constraints θ that can be incorporated by means of a penalty energy

$$E_{\text{strokes}} = \sum_{i \in S_c} w_i((\alpha_i + \beta_i) - \theta_i)^2$$

with w_i being the weight of the constraint as described in Section 4.4. Notice that thanks to the transition functions we can always simply constrain $\varphi_0 = \alpha + \beta$ without worrying about the combinatorial relation between constraints (cf. Figure 4.8(b)). Depending on the application we want to penalize the deviation from orthogonality which in our representation is simply expressed as

$$E_\beta = \sum_i (\beta_i)^2.$$

Thus in total we optimize

$$E_{\text{angle}} = E_{\text{smooth}} + w_{\text{strokes}} E_{\text{strokes}} + w_\beta E_\beta. \tag{4.5}$$

In our application we use a very small weight $w_\beta = 10^{-6}$ to just regularize underdetermined cases, in combination with a weight $w_{\text{strokes}} = 1$ to equally balance smoothness and stroke constraints. Other applications such as quad remeshing may benefit from a higher w_β to favor orthogonal crosses away from constraints.

Greedy mixed-integer optimization. In order to efficiently find good solutions for the mixed-integer problem (4.5) we exploit the following three observations. First, if the integer transition variables p_{ij} are known, E^{ij}_{smooth} is a convex quadratic

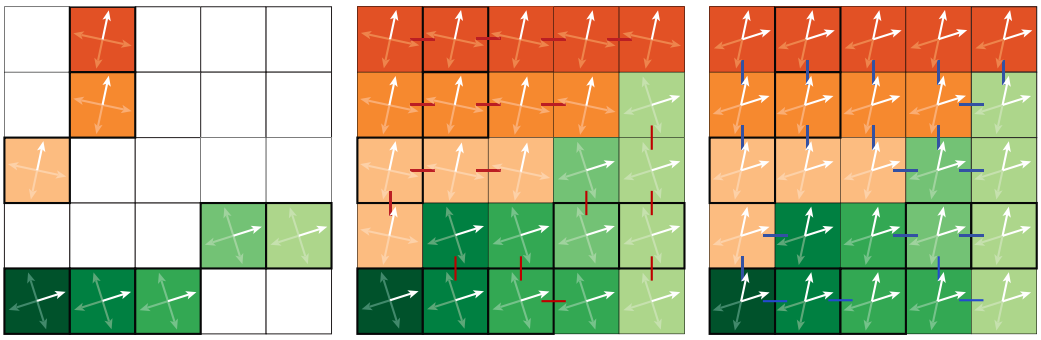
function. We exploit this by solving a series of quadratic problems, which result from the slight modification

$$E_{\text{smooth}} = \sum_i \sum_{j \in \mathcal{N}_i} a_{ij} E^{ij}_{\text{smooth}}$$

with additional boolean variables $a_{ij} \in \{0, 1\}$. Such an a_{ij} is activated, i.e. $a_{ij} = 1$, only if the corresponding p_{ij} is a known constant, while otherwise $a_{ij} = 0$.

We fix the p_{ij} in a greedy order. For each non-activated term we estimate the activation cost as $A_{ij} = \min_{p_{ij}} E^{ij}_{\text{smooth}}$, where this time α and β are kept constant. The best candidate with the smallest cost $\arg\min_{ij} A_{ij}$ is then activated by setting $a_{ij} = 1$ and fixing the corresponding p_{ij}. Subsequently we update the current solution to capture the change due to the newly activated term. We iterate this process until all a_{ij} are activated, i.e. all transition functions are fixed.

Second, determining the p_{ij} that minimizes a A_{ij} can be done by explicitly checking two candidates. Investigating E^{ij}_{smooth} we see that the first term $(\alpha_i + p_{ij}\frac{\pi}{2} - \alpha_j)^2$ is minimized at $p^*_{ij} = \frac{2}{\pi}(\alpha_j - \alpha_i) \in \mathbb{R}$. Since the second term $((-1)^{p_{ij}}\beta_i - \beta_j)^2$ only changes depending on whether p_{ij} is even or odd, we



(a) Stroke constraints (b) Initialization (c) Greedy optimization

Figure 4.11: Main steps of the greedy optimization. Each stroke provides constraints on one of the two representative vectors of a cross (a, outlined pixels). We initialize the cross of each unconstrained pixel to the cross of the closest constraint (b, colors correspond to the closest constraint). We also set $p_{ij} = 0$ between pixels initialized with the same constraint (b, red links). The greedy optimization solves for the remaining p_{ij} (c, blue links) and updates the solution subject to the smoothness energy.

conclude that the optimal value for p_{ij} can only be either $\lceil p^*_{ij} \rceil$ or $\lfloor p^*_{ij} \rfloor$. In the special case of $p^*_{ij} \in \mathbb{Z}$ it is sufficient to check the two candidates $\{p^*_{ij}, p^*_{ij} + 1\}$ since both integer neighbors of p^*_{ij} are equally good, i.e. $A_{ij}(p^*_{ij}+1) = A_{ij}(p^*_{ij}-1)$.

Third, the solution of an unconstrained pixel $i \notin S_c$ is underdetermined if there is no path of activated a_{ij} to one of the constraints. Therefore in order to obtain a unique minimizer, for each pixel we can arbitrarily activate a path of a_{ij} to its closest constraint by fixing the corresponding p_{ij}. This results in a forest, where the root of each spanning tree belongs to a constraint as shown in Figure 4.11(b). Notice that fixing $p_{ij} = 0$ at the spanning tree edges does not restrict the solution space but induces a good initialization for the subsequent greedy integer estimation (Figure 4.11(c)).

Figure 4.6(a) shows an example computation of the greedy mixed-integer optimization. We don't observe undesired singularities in the field, which is a good indicator that our approach effectively avoids local minima. However, the cross field tends to "flatten" away from the strokes, which is a result of optimizing a harmonic energy. We next describe how to optimize for the desired nonlinear energy E_{bend2D}, which better mimics the behavior of 3D curvature lines.

Nonlinear optimization. Since the covariant derivative of E_{bend2D} has a mathematically better behaved expression in vector coefficients, we now switch from the angle representation (α, β) to the vector representation $(\mathbf{u}, \mathbf{v})$. The solution of E_{angle} is used as a starting point for the nonlinear optimization of E_{bend2D}. This refinement is of geometric nature and does not require topological changes. Therefore, we keep the transition functions fixed when discretizing the covariant derivatives with finite differences. The resulting functional can be optimized by a standard Newton method. However, we observed that iterating the optimization of the quadratic approximation

$$(\mathbf{u}^{(i+1)}, \mathbf{v}^{(i+1)}) \leftarrow \min_{u,v} \int_I ||\nabla_{\mathbf{u}} \mathbf{v}^{(i)}||^2 + ||\nabla_{\mathbf{v}} \mathbf{u}^{(i)}||^2 dA$$

where $(\mathbf{u}^{(i)}, \mathbf{v}^{(i)})$ is the solution of iteration i, is sufficient and converges even faster. During this optimization the alignment to strokes is maintained by an additional penalty energy E_{strokes}. We also found that because our smoothness measure

is strongly directional, areas away from the flow induced by the constraints can become unstable. We easily cope with such situations by adding a weak harmonic regularization $\varepsilon_h E_h$ to the functional, resulting in the following energy, which is iteratively optimized

$$E_{\text{vector}} = E_{\text{bend2D}} + w_{\text{strokes}} E_{\text{strokes}} + \varepsilon_h E_h \tag{4.6}$$

where E_{strokes} and E_h are now expressed in vector coefficients instead of angles. Specifically,

$$E_{strokes} = \sum_{i \in S_c} w_i ||\mathbf{u} - (\cos\theta_i, \sin\theta_i)||^2$$

$$E_h = \int_I ||\nabla \mathbf{u}||^2 + ||\nabla \mathbf{v}||^2 dA.$$

We used $\varepsilon_h = 0.1$ for all our results and adjust w_{strokes} according to the sketchiness of the drawing, as discussed in Section 4.9. In practice we use the binary mask provided as input to optimize Equation 4.6 only within the region of interest. On the border of the mask we set Neumann boundary conditions $\nabla \mathbf{u} = 0$ and $\nabla \mathbf{v} = 0$. Interestingly, restricting to the mask does not change the result in the region of interest but significantly speeds up the overall computation. This happens because the optimizer converges only slowly in the unimportant boundary regions, which are far away from sketch constraints, when applied to the whole image.

Figure 4.6c shows that our nonlinear energy effectively improves the curvature lines by inflating the surface parts that appeared to be flat before. This Figure also shows the normal field we obtain after optimizing E_{bend3D}. Figure 4.12 shows the behavior of our method on the sketch of a non-quad patch. The algorithm generates a singularity in the middle of the patch to form a cubic corner.

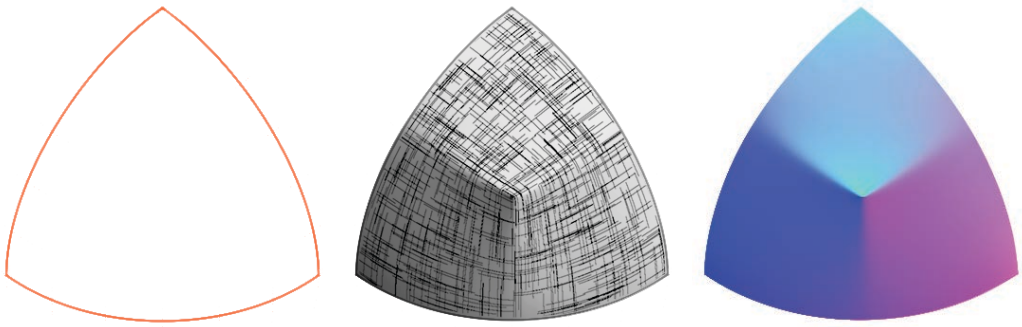


Figure 4.12: This drawing of a non-quad surface patch results in a singularity in the cross field. The corresponding normal field forms a cubic corner.

4.7 Relation to Previous Cross Field Approaches

In this section we clarify the relation of our cross field approach w.r.t. other methods.

Comparison to [Bommes 2009; Bommes 2012]. Our unit-length harmonic optimization is a generalization of [Bommes 2009; Bommes 2012], which is restricted to orthogonal cross fields and relies on continuous relaxation with iterative rounding to solve for integer variables. Nevertheless, our greedy optimizer proceeds in a similar way. In the following we show that for the orthogonal case of $\beta = 0$, the algorithms are identical. The additional difficulty in our case is that the integers p_{ij} contribute nonlinearly due to expressions $(-1)^{p_{ij}}$, which additionally make continuous relaxation impossible without switching to complex numbers. Thus, instead of optimizing the continuous relaxation, we deactivate all terms with unknown p_{ij}. At a first glance this appears suboptimal compared to the continuous relaxation. However, by considering that in the orthogonal case each p_{ij} contributes only to one term $(\alpha_i + p_{ij}\frac{\pi}{2} - \alpha_j)^2$, we see that the continuously relaxed terms also always vanish by the choice $p_{ij} = 2/\pi(\alpha_j - \alpha_i)$. Thus, for $\beta = 0$ our greedy approach based on activation variables is identical to the relaxation method of [Bommes 2009; Bommes 2012]. Consequently we also benefit from all performance optimizations that were proposed in [Bommes 2012], including a hierarchy of solvers and

multiple activations.

Comparison to CDFs. Since the handling of non-orthogonal cross fields in the context of conjugate direction fields (CDFs) [Liu 2011] is related to our approach, we discuss the differences in more detail. Similarly to us, Liu et al. use an angle-based representation (θ, α), which in our notation corresponds to $\theta = \varphi_0$ and $\alpha = \varphi_1 - \varphi_0$. The transition functions are handled by $p_1, p_2 \in \mathbb{Z}$ and $q \in \{0, 1\}$ while in our case a single $p \in \mathbb{Z}$ is sufficient. Apart from these notational subtleties, which mostly affect the simplicity of formulas, the most important difference is the chosen smoothness measure and the corresponding optimization strategy. The smoothness measure of [Liu 2011] exploits the periodicity of the $\cos$ function to eliminate all integer degrees of freedom (DOFs) and it is shown that the resulting functional is a generalization of the one proposed in [Hertzmann 2000] for orthogonal cross fields. As discussed in [Liu 2011] these nonlinear functionals induce a tendency to end up in local minima with unsatisfactory additional singularities (cf. Figure 4 in [Liu 2011]), even in case of a good initialization.

On the contrary our smoothness measure contains integer DOFs and is a generalization of Bommes et al. [Bommes 2009]. Since every step in the greedy integer estimation solves a simple linear problem, similarly to [Bommes 2009], unsatisfactory additional singularities are effectively prevented. Consequently, we believe that our novel non-orthogonal cross field representation and optimization is a valuable general tool with numerous applications apart from concept sketching, such as surface meshing.

4.8 Additional Details

User interface. The accompanying video illustrates a typical interactive session with our tool. Users first load an existing bitmap sketch and its mask and paint over curvature and discontinuity strokes in different colors. Design literature explains that "cross-sections on a surface explain or emphasize its curvature" [Eissen 2008], which suggests that designers know the difference between lines that convey curvature and other discontinuity lines. In addition to the stroke annotations, we also ask users to select one of the two possible consistent solutions

of each surface patch normal field (Section 4.5.3, Figure 4.13). We implemented these user indications as unary constraints in the labeling problem (Appendix). In practice, several indications are sometimes necessary to obtain a consistent result over complex patches. Finally, we also provide users the ability to combine layers when sketching complex objects made of independent parts.

Pixels on discontinuity strokes do not constrain the orientation and smoothness of the cross field and are also not considered when solving for the globally consistent 3D solution. As a result, regions bounded by discontinuity strokes form isolated patches in the solution and discontinuity strokes do not receive values. We assign normals to discontinuity pixels as a post-process by diffusing nearby normals [Winnemöller 2009].

Sketch pre-processing. We apply a few simple image processing operations on the input image before running our algorithm. We found that applying a small Gaussian blur to remove noise produces a more accurate estimation of local orientations. We also observe that artists draw strokes with varying strength, darker strokes denoting more confidence. We apply a permissive threshold to select dark and lighter strokes (0.8 in our implementation). The constraint weight w_i then implicitly accounts for the strength of the strokes as darker ones have a

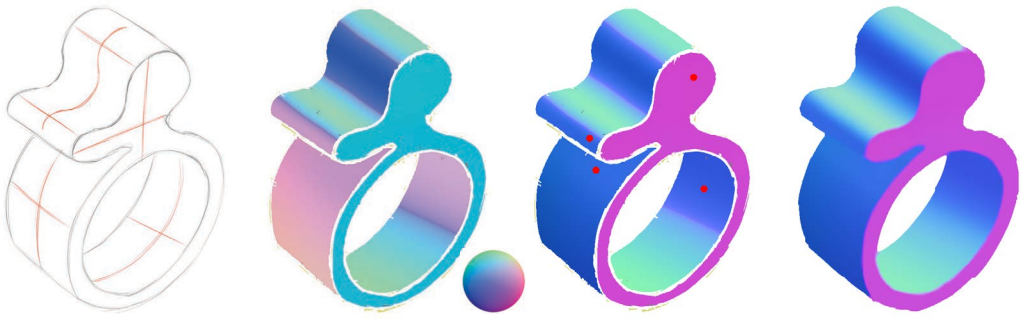


Figure 4.13: Curvature lines can be interpreted as running over a convex or a concave surface patch. Based on our local guess (b), the user clicks on inconsistent patches (c, red dots) to flip their orientation. The local guess is then refined by the 3D BendField energy (d).

smaller minor eigenvalue.

Texture Mapping. For texture mapping we adapt the parametrization step of [Bommes 2009] to non-orthogonal cross fields with anisotropic sizing. In order to get the desired parametrization (s, t) of our surface we optimize

$$E_{\text{param}} = \int_I \left([\nabla s, \nabla t]^T [\mathbf{u}, \mathbf{v}] - I_2 \right)^2 dA$$

with I_2 being the two dimensional identity matrix and

$$[\nabla s, \nabla t]^T = \begin{pmatrix} \partial s/\partial x & \partial s/\partial y \\ \partial t/\partial x & \partial t/\partial y \end{pmatrix}, \; [\mathbf{u}, \mathbf{v}] = \begin{pmatrix} u_x & v_x \\ u_y & v_y \end{pmatrix}$$

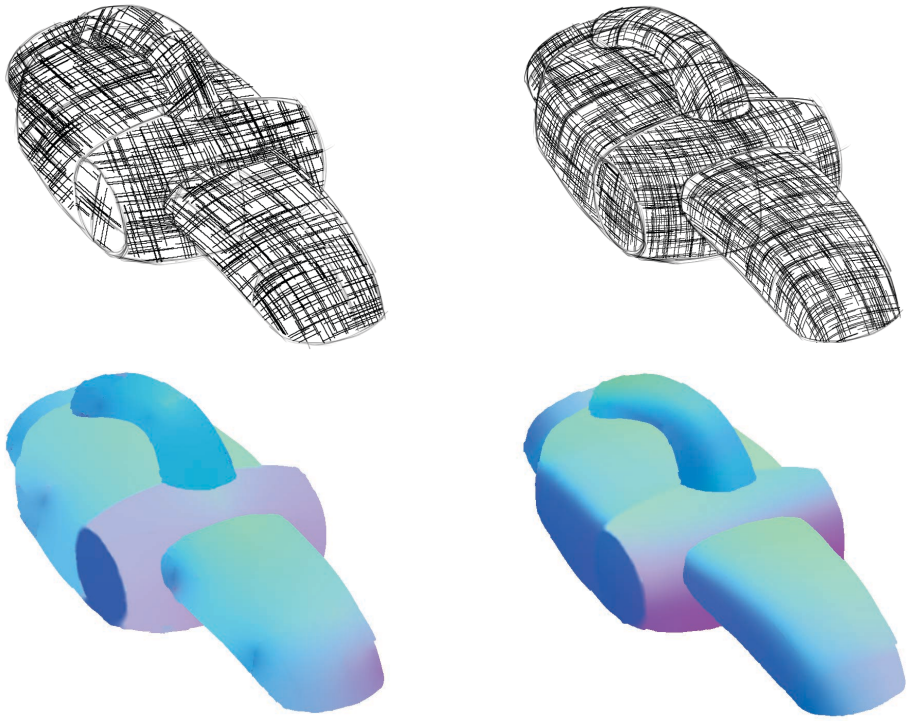
being the differential which transforms vectors from image space to texture space. Intuitively this energy states how well the given $\mathbf{u}, \mathbf{v}$ vectors map to the Cartesian s, t axes in texture space. Thus the inverse mapping tries to aligns the texture with the non-orthogonal cross field. The length distortion due to foreshortening is encoded into the length of $\mathbf{u}$ and $\mathbf{v}$. While our cross field is not guaranteed to be integrable, we didn't observe significant distortions of the parameterization in practice.

Optimization. For the greedy mixed-integer optimization we use a hierarchy of solvers, i.e. local Gauss Seidel, Conjugate Gradient and Sparse Cholesky, as explained in [Bommes 2012]. The Conjugate Gradient and Sparse Cholesky is taken from Eigen3 [Guennebaud 2010], which is also used for the optimization of all quadratic energy functionals. For the nonlinear optimization of Section 4.5.3 we apply the interior point method of IPOPT [Wächter 2006]. We provide source code as supplemental materials to facilitate reproduction.

Table 4.1 details the time spent by our implementation on each step of the optimization, for two sketches. The current bottleneck resides in the nonlinear optimization of the 2D and 3D BendField energies (Equations 4.4 and 4.6). However, we note that our cross fields are piece-wise smooth, which makes our problem an ideal candidate for hierarchical algorithms like multigrid [Briggs 2000], although care should be taken to properly handle transition functions between levels of the hierarchy [Bommes 2013a].

Sketch	Resolution	Harmonic	2D BendField	3D BendField
Kettle	900 × 800	1 min. 30 sec.	32 min.	9 min.
Vacuum	900 × 700	2 min. 17 sec.	21 min.	24 min.

Table 4.1: Detailed timing for the main steps of our method for a simple sketch (water kettle, Figure 4.17) and a complex one (vacuum cleaner, Figure 4.1). We used 10 iterations to optimize the 2D BendField with Eqn. (4.6), although we observed that the energy decreases quickly during the first 3 iterations, then converges to a plateau value.



(a) Harmonic energy (b) Our energy

Figure 4.14: Comparison between a harmonic energy and our energy on the same input as Figure 4.1. A harmonic energy tends to flatten the shape away from user strokes, producing "tent" artifacts at stroke intersections (a). Our energy better captures the bending of the surface (b). We used the same weights for the two versions of the algorithm.

4.9 Results and Evaluation

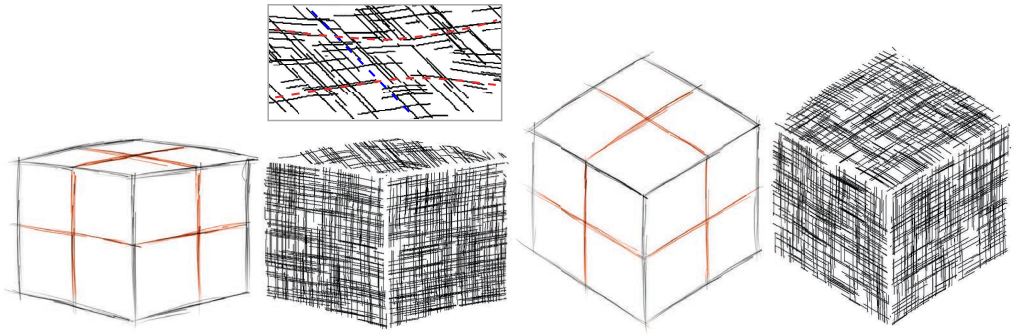
Figure 4.1, 4.3, 4.13, 4.16, 4.17, 4.21, 4.20 illustrate the results of our method on a range of concept sketches. A major advantage of our method is its ability to pro-

cess existing sketches. We produced all our inputs by reproducing drawings from design books and websites that are representative of the distortions and inaccuracy found in real sketches. We reproduced these drawings to avoid copyright issues and to remove decorative lines (cross-hatching and texture) that users would not draw in our context. We provide links to the original drawings as supplemental material. In theory, our algorithm requires at least two intersecting curvature lines per isolated surface patch, although users can draw more lines to refine bending over complex surfaces.

We visualize our cross-fields with hatching [Hertzmann 2000] and provide a color-coded visualization of the surface normals estimated by our algorithm. The cross-fields and normals provide a vivid sense of the 3D shapes depicted by the sketches, capturing the overall orientation of the surfaces as well as subtle bending, such as the folding shape of the stool (Figure 4.3) and the wavy handle of the bag (Figure 4.21). Figure 4.21 additionally shows the use of our normals and cross-fields for shading and texture mapping. Figure 4.14 shows how a harmonic smoothness energy does not capture surface bending as well as our BendField energy.

Limitations. Because our approach works on bitmaps, the accuracy of the result is dependent of the image resolution. Fine details, such as the thin legs of the chair in Figure 4.21, are not well captured by the structure tensor. While we added these details with decorative strokes, an alternative would have been to sketch them at higher resolution in a separate layer and then composite them in the final image.

Our algorithm infers the directions of the cross-field from the local orientation of the strokes. As a result, our approach sometimes fails to distinguish a strongly foreshortened crossing between two lines from a bend on a single line, since the two configurations locally form a similar wide angle. In theory, our algorithm will always interpret intersecting lines as two different directions if they deviate by less than 45° from orthogonality, while greater deviations can be compensated for by the transition function and be interpreted as a bending over a single direction. In practice the global configuration of the cross-field



(a) Too foreshortened (b) Proper foreshortening

Figure 4.15: In the presence of strong foreshortening, such as on the top face of the cube in (a), intersecting curvature lines form the same local configuration as overlapping strokes on a curvy line. As a result, our algorithm interpret these lines as constraints on only one of the two directions of the cross-field. Designers usually avoid strong foreshortening to maximize information, providing proper constraints for our algorithm (b).

can sometimes result in angles that deviate more than this lower bound. Figure 4.15(a) illustrates the behavior of our algorithm on a strongly foreshortened cube, where the curvature lines on the top face are interpreted as constraints on one of the two directions of the cross-field, the other direction being free. Figure 4.15(b) shows a cube from a less foreshortened view where the crossing lines form an angle closer to 90° and as such are better captured by our approach. Fortunately, designers are trained to draw objects from *informative viewpoints* that minimize foreshortening over most surfaces [Eissen 2011; Shao 2012], as demonstrated by our results over typical sketches.

Effect of parameters. Figure 4.16 illustrates the effect of the two main parameters of our algorithm. These parameters offer a trade-off between fidelity to the input strokes and smoothness of the solution. In particular, sketchy lines can result in wiggles in the normal field, which can be removed by increasing the spatial extent σ_s of the bilateral filter during orientation estimation (Section 4.4) and by reducing the constraint weight w_{strokes} in Equation 4.6. However, too much filtering can result in a loss of detail, while too much smoothing tends to flatten the surface. We used the same preset of $\sigma_s = 13$ and $w_{\text{strokes}} = 0.25$ for most sketches, except for the very sketchy drawings (Figure 4.3 and 4.17) for which we

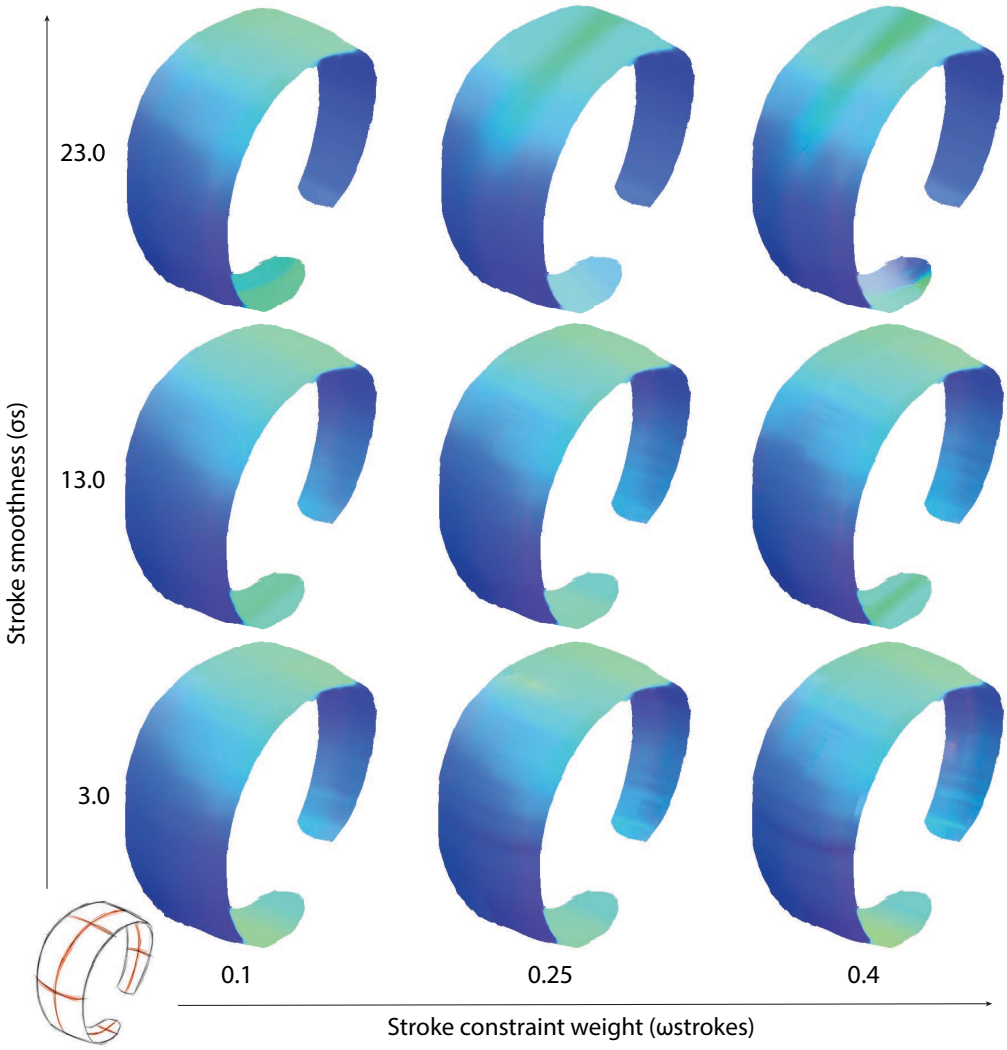


Figure 4.16: The stroke smoothness and constrain weights offer a trade-off between fidelity to the input drawing and smoothness of the normal field. While a range of parameters produce similar results, too much smoothing removes details and flatten the surface (top left corner), while too strong constraints produce wiggles because of the sketchy strokes (right column).

use $\sigma_s = 23$ and the clean CrossShade curves (Figure 4.20) for which we used $w_{\text{strokes}} = 0.1$.

Robustness to sketchy lines. We designed our method to be robust to the sketchy lines typical of concept drawings. Figure 4.17 evaluates this robustness

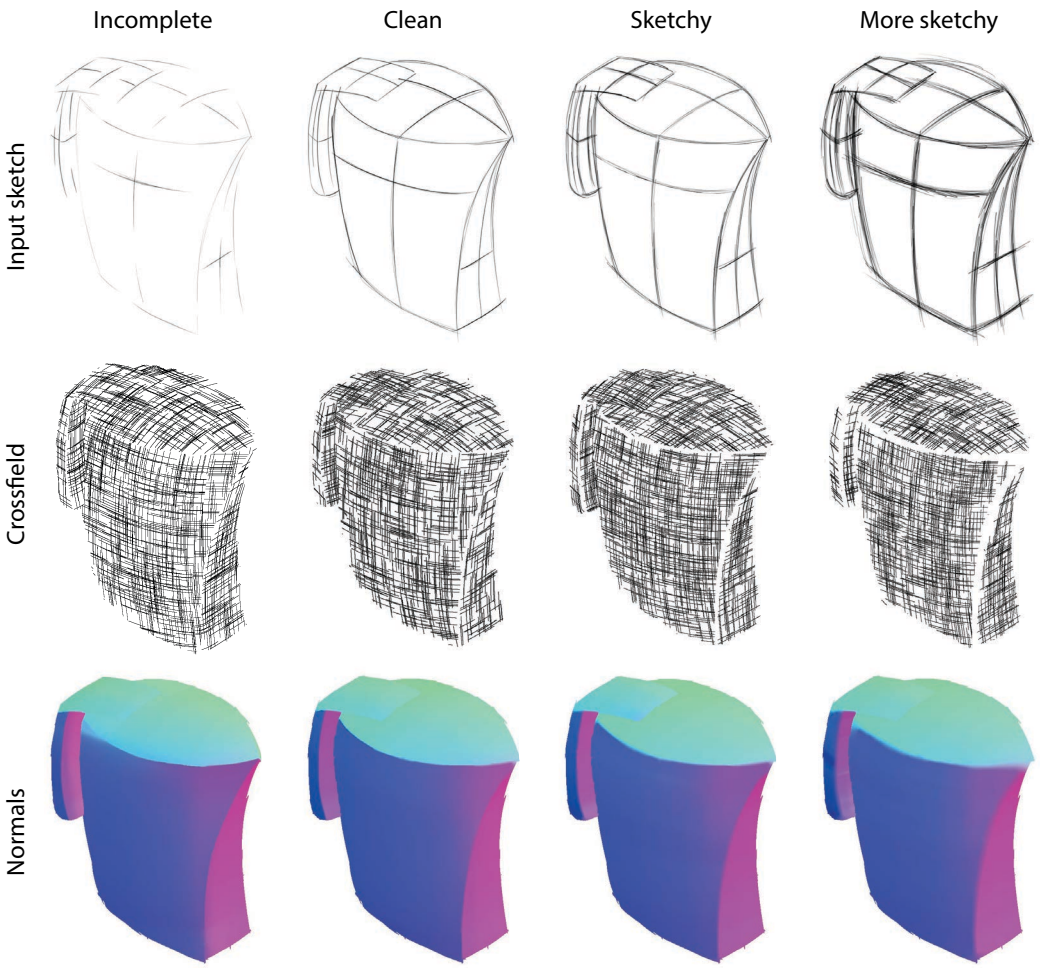


Figure 4.17: Our approach is robust to different levels of sketchiness, from sparse strokes with holes (left) to many overlapping strokes (right). Despite these drastic differences in input style, our method produces consistent cross-fields and normals.

on four versions of the same sketch, with an increasing density of strokes. Our method produces similar cross-fields and normals for the various levels of sketchiness, although fine details are lost for very sketchy drawings. While our scattered interpolation handles sparse and incomplete curvature constraints, holes in discontinuity lines can result in smooth transitions across surface patches.

Comparison to ground truth. We derived our BendField energy from properties of curvature lines. Figure 4.18 compares our interpolated cross fields to ground truth curvature lines generated from 3D surfaces. We chose these surfaces

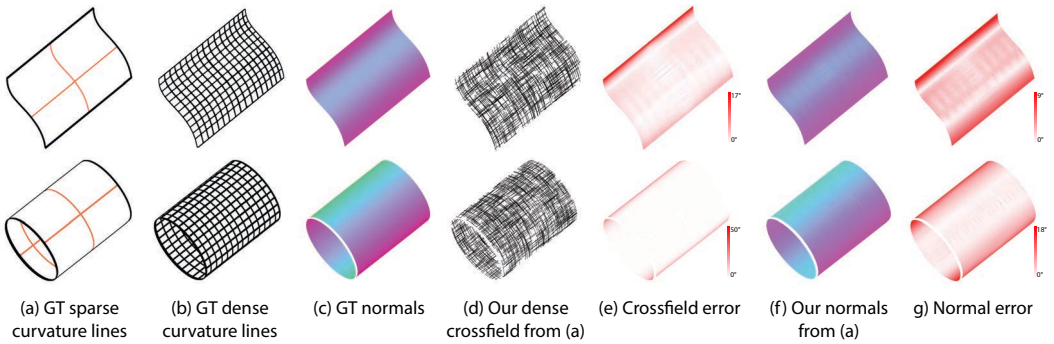
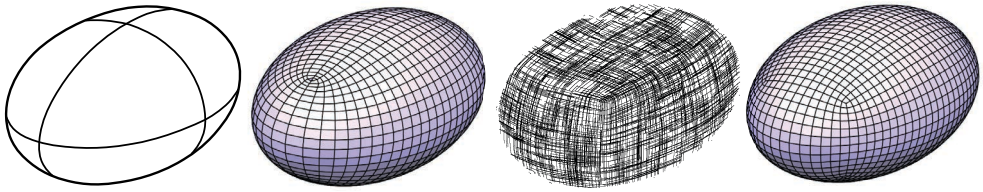


Figure 4.18: Comparison with ground truth curvature lines and normals. Our cross field aligns with the ground truth dense curvature field (b,d), producing normals that closely match the ones of the ground truth surface (c,e). Our cross field and normals are less accurate near boundaries and silhouettes where junctions make the estimation of orientation of curvature lines less accurate and regularization penalizes strong foreshortening and non-orthogonality. The cylinder has a mean error on cross field directions of $2.86°$ and standard deviation of $5.43°$ and on normals of $1.26°$ and standard deviation of $2.4°$, the wave has a mean error of $2.00°$ and standard deviation of $2.43°$ on directions and of $0.85°$ and standard deviation of $1.64°$ on normals.

to have no umbilical points and to be perfect minimizers of the BendFields energy since their lines of curvature are also geodesics. We applied our complete pipeline to the rasterised sparse curvature lines using a small stroke smoothness $\sigma_s = 3$ and $w_{\text{strokes}} = 0.1$.

Our cross-field closely matches the projected curvature lines with a mean error of 2.49 degrees on the directions (standard deviation of 4.52 degrees), resulting in visually similar normals with a mean error of less than 1.12 degrees (standard deviation of 2.09 degrees). Small wiggles are noticeable in the normal field, which can be removed by increasing σ_s at the cost of flattening the shape. We provide as supplemental materials the results of the same experiment using ground truth 2D constraints as input to bypass the initial estimation of stroke orientation. The errors in this experiment are lower than when running the complete pipeline, yet distributed similarly with a mean error of 1.87 degrees on the cross field directions (standard deviation of 3.94 degrees) and 1.06 degrees on the normals (standard deviation of 1.97 degrees). Most errors occur near discontinuity lines and silhouettes where the estimation of local orientation is less accurate and our regularizers



(a) GT sparse curvature lines (b) GT dense curvature lines (c) Our dense cross field from (a) (d) Surface cross field [Bommes et al. 09]

Figure 4.19: Comparison with ground truth curvature lines in the presence of umbilical points. Our regularized cross field deviates from the non-geodesic curvature lines (b,c) and positions singularities similarly to the surface cross field of [Bommes 2009] (d).

penalize strong foreshortening and non-orthogonal crosses (Equation 4.4 and 4.5). An interesting direction for future research would consist in combining our approach with inflation methods [Johnston 2002] in order to leverage both the 3D cues provided by curvature lines and smooth silhouettes.

Figure 4.19 provides an evaluation against a more complex surface with umbilical points. Our regularized cross field positions a singularity in the center of the triangular face of the sketch, while the ground truth singular points lie on the great circles of the ellipsoid. Figure 4.19(d) shows that our algorithm actually behaves similarly to the surface cross field algorithm of Bommes et al. [Bommes 2009], which favors smooth geodesic curves away from regions with high anisotropic curvature.

Comparison to prior work. Figure 4.20 provides a comparison of our normal fields and shading with the ones generated by the CrossShade algorithm [Shao 2012]. While both algorithms estimate normals by leveraging orthogonality of curvature lines, they target different input and perform data interpolation in a different order. CrossShade requires clean vectorial curves as input, which provide a high degree of precision and smoothness. In contrast, our method processes bitmap drawings with a finite resolution and sketchy lines. From an algorithmic point of view, CrossShade estimates normals solely at curve intersections and propagates these estimates along and in-between curves using parametric Coons patches. Our algorithm operates in a different order, first applying scattered interpolation on the strokes to form a dense curvature field and then

estimating normals at each pixel. CrossShade also enforces planar cross-section curves, which our local formulation cannot do. In practice, our algorithm tends to produce a flatter result near boundaries of curved surface patches, such as on the lens of the camera in Figure 4.20, where the junctions make the orientation estimation less reliable (Section 4.4). We plan to explore the estimation of two directions near junctions to address this issue [Aach 2006]. Nevertheless, our approach produces results visually similar to CrossShade, without requiring users to be familiar with vector drawing tools.

4.10 Conclusion

Sketch-based modeling systems traditionally take clean vectorial curves as input. In this Chapter we have explored an alternative approach by extrapolating curvature lines from bitmap drawings. Our approach relies on a scattered-data interpolation to be robust to the rough drawings common in concept sketching. The resulting 2.5D cross fields, which we call Bend Fields, allow a range of sketch-editing applications originally developed for 3D surfaces, such as local shading, texture mapping and cross-hatching.

(a) Rasterized CrossShade curves (b) Our normals (c) Our shading (d) CrossShade normals (e) CrossShade shading

Figure 4.20: Comparison with CrossShade [Shao 2012]. Our method produces qualitatively similar results without the need for vectorial curves. Note that we rasterized the CrossShade curves as polylines and did no attempt to remove extraneous dangling segments at extremities.

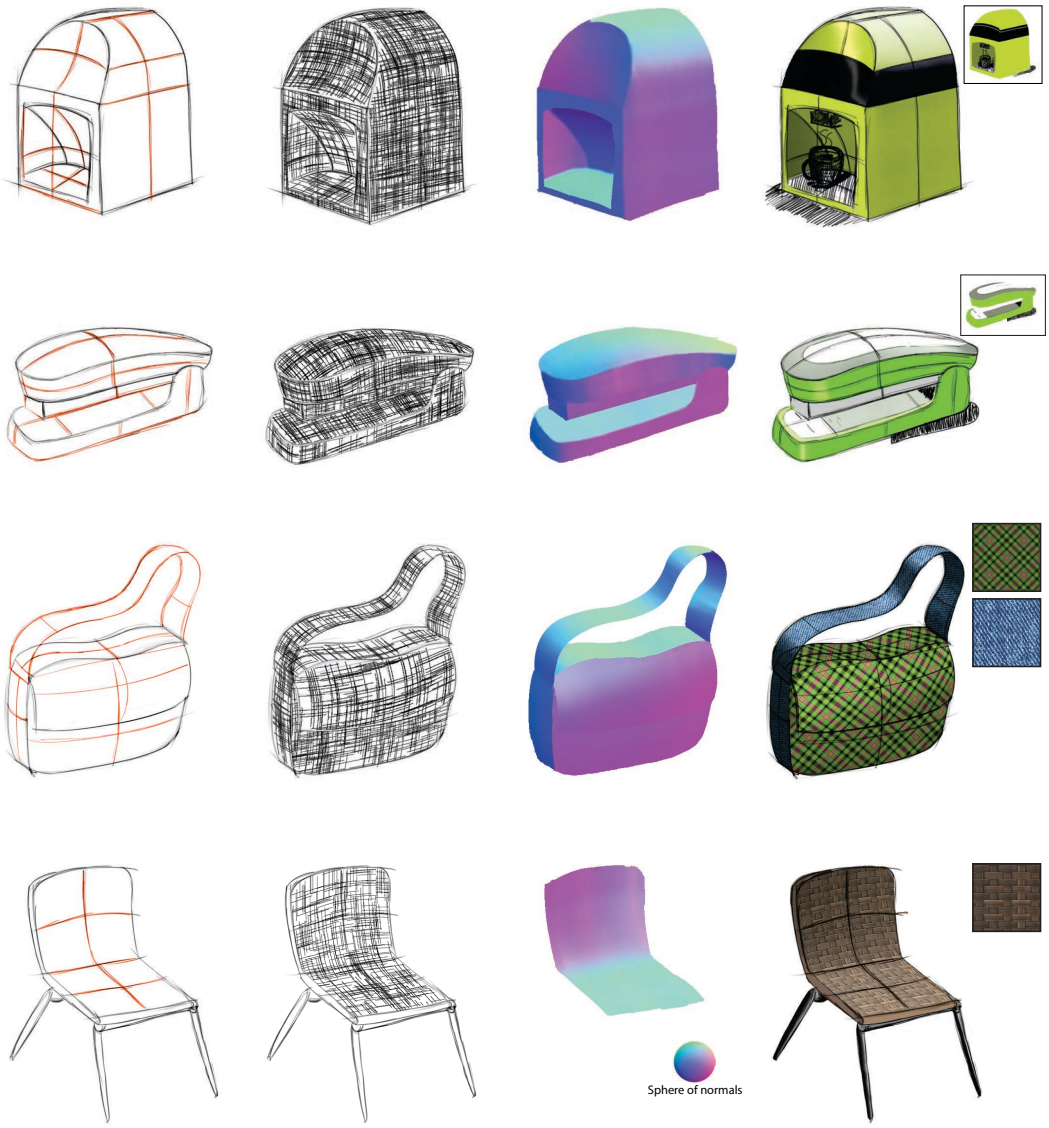


Figure 4.21: Cross-fields and normals generated by our method for a coffee machine, a stapler, a bag and a chair. The bag and chair also show the use of texture mapping.

Appendix

We describe in this appendix our implementation of the binary-labeling problem to find a consistent orientation of directions over a surface patch. We express the image as a graph, where each pixel is a node connected to its upper and right neighbor

by an edge. Each node i stores two candidate values $u_z(i)$ and $-u_z(i)$, and each edge (i, j) stores the 2×2 pairwise cost matrix of assigning each possible pair of values to the nodes i and j. We express these costs as the absolute difference between the two values. Our goal is to select the value of each node that minimize the cost over all edges, which we solve using the *Convergent Tree-reweighted Message Passing* algorithm[1] [Kolmogorov 2006]. We optionally provide users the ability to constrain one of the two values at a pixel, which the algorithm then propagates to other pixels. This feature is particularly useful to resolve the ambiguity between convex and concave surface patches, which have opposite orientations. We express these constraints as a unary penalty term that we set to 0 for the solution we want to favor and to 1 for the solution we want to penalize. We also account for these constraints in the pairwise terms that we set to 1 for the solution we want to penalize. Finally, care should be taken to properly handle the transition functions when computing the pairwise term.

[1] Implementation available at http://research.microsoft.com/en-us/downloads/dad6c31e-2c04-471f-b724-ded18bf70fe3/

CHAPTER 5

Conclusion and Future Work

Drawing is a fundamental tool for different activities in art and industry, yet still a challenge for all kind of practitioners. Our work has focused on facilitating and accelerating drawing for learners as well as hobbyists and professional designers.

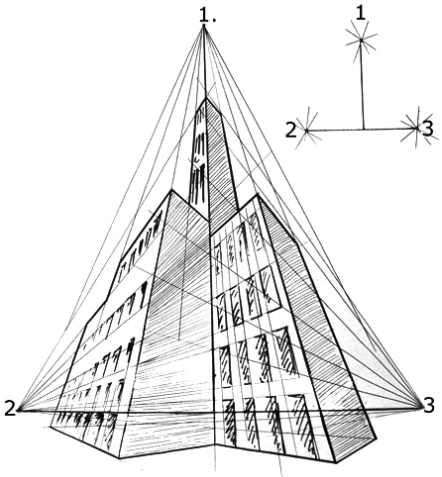
Along this Thesis we have followed a common methodology. First we formalize the drawing principles applied by artists and designers. Such is the case in Chapter 2, where we identified a set of construction lines employed by artists to guide observational drawing. We also distilled the principles applied in metal wire cutting for jewelry fabrication in Chapter 3, and we identified the assumptions behind curvature lines in sketches in Chapter 4.

Given these principles, we then proposed computer tools that can assist or perform design tasks for users. Our drawing assistant automatically computes construction lines from a model photograph, and guides the user in drawing it. The algorithm proposed in Chapter 3 can automatically find a sketch segmentation suitable for metal wire fabrication. Finally, our BendField algorithm from Chapter 4 can recover the intended 3D shape of a sketch to compute shading and texturing.

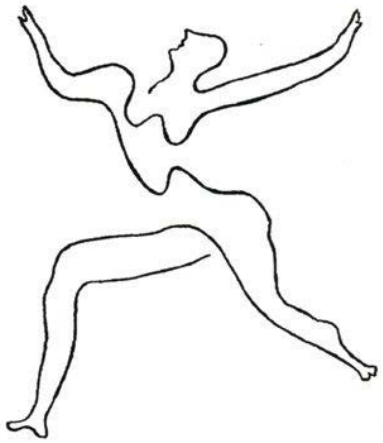
5.1 Short term goals

The nature of problems we addressed made our solutions diverse. For this reason, this Thesis opens a large number of potential research directions. We now briefly describe some of these directions:

Assisting domain-specific drawing techniques. While in Chapter 2 we focused on *generic* techniques for observational drawing, *domain-specific* techniques could also be integrated into our system. For example, computer vision algorithms can detect vanishing points that form the basis of perspective drawing for architecture (Figure 5.1a). Our approach could also be extended with augmented reality technology, enabling users to practice drawing with real subjects.



(a) Vanishing points in perspective drawing



(b) Line drawing by Picasso

Figure 5.1: Specific drawing techniques like perspective with vanishing points are widely used in architecture drawing (a) (source: http://aliar.iics-k12.com). Picasso's abstracted line drawings capture the essence of a shape with a single stroke (b).

Proposing new shape abstraction algorithms. The segmentation algorithm proposed in Chapter 3 finds Eulerian paths that *exactly* reproduce the input drawing, up to optional bridges. However, part of the art of wire wrapping involves *abstracting* a shape to make it more suitable for fabrication with smooth wires. Figure 5.1b illustrates the kind of abstraction required, where the challenge is to preserve the essence of the shape, while accounting for fabrication and aesthetic constraints of jewelry making (single wire, smooth curves).

(a) Input (b) Heightfield from different viewpoints

Figure 5.2: Our normal-fields from Chapter 4 can be integrated to form height-fields [Nehab 2005]. Here we reconstructed each layer separately and positioned them in depth manually. While the resulting surfaces suggest the potential of our approach for 3D sketch-based modeling, additional work is needed to deal with hidden parts of the model and to correct for distortions due to perspective and sketch inaccuracy.

Improving 3D sketch-based modeling from rough sketches. While our normal-fields from Chapter 4 can be integrated to form height-fields (Figure 5.2), the resulting surface is often distorted due to perspective inaccuracy in sketches [Schmidt 2009a]. An interesting direction of research would be to detect and enforce regularity constraints over the cross field, such as symmetry, to correct for these distortions [Xu 2014]. Recovering the full intended surface from a rough sketch would drastically shorten the 3D modelling pipeline, by alleviating the need for clean vector drawing as input.

5.2 Long term goals

In the long term, we would like to apply the introduced techniques and our expertise to the field of architectural design. In the future, we want to focus on three main directions:

Architectural sketch interpretation. As any design domain, architecture has its own set of principles that can be formalised in order to automate design tasks.

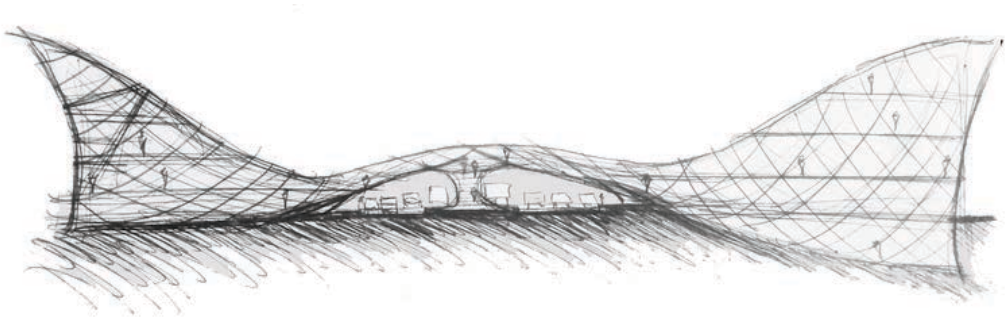


Figure 5.3: Quick concept sketch from Wellington's train station (source: http://franobazalo.weebly.com).

For example, the sketch in Figure 5.3 represents a freeform building, where the hatching indicates a ceiling made of quadrangular glass panels. Our cross fields could help to extract and interpret the paneling structure from the sketch. However, new principles raise new challenges. Sketches in architecture are extremely rough and usually highly foreshortened due to extreme perspectives. To make interpretation possible, we would also need to integrate other visual cues such as vanishing lines (Figure 5.1a).

Figure 5.4: "Maquette Pour une Structure de Rencontre et de Réflexion". Wire model by Antti Lovag (source: http://www.bdonline.co.uk).

Architectural modelmaking. As a complement to 2D sketches, architects generally build physical models of their designs in order to better understand shape and communicate with others. We would like to tackle the challenges faced by modelmakers when constructing these models. In particular, we believe that the computational design of jigs and other intermediate support structures has a great potential for wire or clay model crafting (Figure 5.4). This direction of research relates to recent work on optimizing the placement of support chains for the assembly of self-supporting structures [Deuss 2014].

Architectural manufacturing. New materials engender new forms. With the increasing variety of construction materials, modern architecture has become more complex. Freeform architecture raises new challenges not only for modelling, but also for final construction. These structures are sometimes not feasible due to physical or financial constraints (Figure 5.5). Similarly to how we decomposed a shape into wires suitable for jewelry making, 3D structures can be decomposed into parts that satisfy manufacturing and financial constraints [Eigensatz 2010]. Another promising direction is to propose design tools that directly constraint the spectrum of possible shapes [Yang 2011].

Figure 5.5: The versatility of concept design is restricted only by imagination. Yet, budget and structural integrity have to be considered for construction (source: https://taboodada.wordpress.com/).

5.3 Concluding remarks

Despite the ubiquity of pen and paper, drawing has been restricted to a trained elite. We strongly believe that with automation and guidance, digital tools have the potential to make drawing and design accessible to all. The three projects described in this Thesis represent steps towards that direction, not only helping professionals, but also hobbyists and beginners.

Bibliography

[Aach 2006] Til Aach, Cicero Mota, Ingo Stuke, Matthias Muhlich and Erhardt Barth. *Analysis of superimposed oriented patterns*. IEEE Trans. on Image Processing, vol. 15, no. 12, pages 3690–3700, 2006. (Cited on pages 60 and 92.)

[Agrawala 2003] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan and Barbara Tversky. *Designing Effective Step-by-step Assembly Instructions*. ACM Transactions on Graphics (Proc. of SIGGRAPH), vol. 22, no. 3, pages 828–837, July 2003. (Cited on page 38.)

[Agrawala 2011] Maneesh Agrawala, Wilmot Li and Floraine Berthouzoz. *Design principles for visual communication*. Communications of the ACM, vol. 54, no. 4, pages 60–69, 2011. (Cited on page 4.)

[Alliez 2003] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy and Mathieu Desbrun. *Anisotropic Polygonal Remeshing*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 22, no. 3, 2003. (Cited on page 60.)

[Arbelaez 2011] P. Arbelaez, M. Maire, C. Fowlkes and J. Malik. *Contour detection and hierarchical image segmentation*. IEEE Trans. on Pattern Analysis and Machine Intelligence, no. 99, pages 1–1, 2011. (Cited on page 18.)

[Bae 2008] S.H. Bae, Ravin Balakrishnan and Karan Singh. *ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models*. In Proc. User Interface Software and Technology (UIST), 2008. (Cited on pages 56 and 57.)

[Bae 2009] Seok-Hyung Bae, Ravin Balakrishnan and Karan Singh. *EverybodyLovesSketch: 3D sketching for a broader audience*. In ACM Symp. on User Interface Software and Technology (UIST), pages 59–68, 2009. (Cited on page 12.)

[Bartolo 2007] A. Bartolo, K. P. Camilleri, S. G. Fabri, J. C. Borg and P. J. Farrugia. *Scribbles to Vectors: Preparation of Scribble Drawings for CAD Interpretation*. In Proc. of Eurographics Workshop on Sketch-based Interfaces and Modeling (SBIM), pages 123–130. ACM, 2007. (Cited on page 59.)

[Belongie 2002] S. Belongie, J. Malik and J. Puzicha. *Shape matching and object recognition using shape contexts*. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pages 509–522, 2002. (Cited on pages 21 and 22.)

[Bessmeltsev 2012] Mikhail Bessmeltsev, Caoyu Wang, Alla Sheffer and Karan Singh. *Design-driven quadrangulation of closed 3D curves*. ACM Transactions on Graphics (Proc. SIGGRAPH Asia), vol. 31, no. 6, page 178, 2012. (Cited on pages 62, 66 and 69.)

[Biard 2010] Luc Biard, Rida T. Farouki and Nicolas Szafran. *Construction of rational surface patches bounded by lines of curvature*. Computer Aided Geometric Design, vol. 27, pages 359–371, 2010. (Cited on pages 66 and 67.)

[Blum 1967] H. Blum*et al.* *A transformation for extracting new descriptors of shape*. Models for the perception of speech and visual form, vol. 19, no. 5, pages 362–380, 1967. (Cited on page 18.)

[Bommes 2009] David Bommes, Henrik Zimmer and Leif Kobbelt. *Mixed-integer quadrangulation*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 28, no. 3, page 77, 2009. (Cited on pages 56, 61, 70, 74, 75, 76, 81, 82, 84 and 91.)

[Bommes 2012] David Bommes, Henrik Zimmer and Leif Kobbelt. *Practical mixed-integer optimization for geometry processing*. In Proceedings of the 7th international conference on Curves and Surfaces, pages 193–206, Berlin, Heidelberg, 2012. Springer-Verlag. (Cited on pages 81 and 84.)

[Bommes 2013a] David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez and Leif Kobbelt. *Integer-grid Maps for Reliable Quad Meshing*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 32, no. 4, pages 98:1–98:12, July 2013. (Cited on page 84.)

[Bommes 2013b] David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini and Denis Zorin. *Quad-Mesh Generation and Processing: A Survey*. Computer Graphics Forum, vol. 32, no. 6, pages 51–76, 2013. (Cited on pages 60 and 69.)

[Bradley 2003] B. Bradley. Drawing people: how to portray the clothed figure. North Light Books, 2003. (Cited on pages 9, 15, 16, 17, 19 and 20.)

[Briggs 2000] William L. Briggs, Van Emden Henson and Steve F. McCormick. A multigrid tutorial (2nd ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. (Cited on page 84.)

[Chi 2012] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li and Björn Hartmann. *MixT: automatic generation of step-by-step mixed media tutorials*. In ACM Symp. on User Interface Software and Technology (UIST), 2012. (Cited on page 12.)

[Cole 2012] Forrester Cole, Phillip Isola, William T. Freeman, FrÃ©do Durand and Edward H. Adelson. *Shapecollage: Occlusion-Aware, Example-Based Shape Interpretation.* In European Conference on Computer Vision (ECCV), pages 665–678. Springer, 2012. (Cited on page 58.)

[Coros 2013] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik and Bernd Bickel. *Computational Design of Mechanical Characters*. ACM Transactions on Graphics (Proc. of SIGGRAPH), vol. 32, no. 4, pages 83:1–83:12, July 2013. (Cited on page 38.)

[Cummmings 2012] Danielle Cummmings, Francisco Vides and Tracy Hammond. *I don't believe my eyes!: geometric sketch recognition for a computer art tutorial*. In Proc. International Symposium on Sketch-Based Interfaces and Modeling, pages 97–106, 2012. (Cited on page 12.)

[DeField 2015] Arla DeField. Make wire wrap jewelry: Basic wire wrapping techniques and jewelry tutorials. 2015. (Cited on page 39.)

[Denning 2011] Jonathan D. Denning, William B. Kerr and Fabio Pellacini. *MeshFlow: interactive visualization of mesh construction sequences*. ACM Trans. on Graphics (Proc. SIGGRAPH), vol. 30, no. 4, 2011. (Cited on page 12.)

[Deuss 2014] Mario Deuss, Daniele Panozzo, Emily Whiting, Yang Liu, Philippe Block, Olga Sokrine-Hornung and Mark Pauly. *Assembling Self-Supporting Structures*. ACM Transactions on Graphics, vol. 33, no. EPFL-ARTICLE-201940, 2014. (Cited on pages 39 and 99.)

[Diamanti 2014] Olga Diamanti, Amir Vaxman, Daniele Panozzo and Olga Sorkine-Hornung. *Designing N-PolyVector Fields with Complex Polynomials*. Computer Graphics Forum (proceedings of EUROGRAPHICS Symposium on Geometry Processing), vol. 33, no. 5, 2014. (Cited on page 61.)

[Dismore 2011] Heather Dismore. Jewelry making & beading for dummies. –For dummies. Wiley, 2011. (Cited on page 39.)

[Dixon 2010] D. Dixon, M. Prasad and T. Hammond. *iCanDraw: Using sketch recognition and corrective feedback to assist a user in drawing human faces*. In Proc. of the International Conference on Human Factors in Computing Systems (CHI). ACM, 2010. (Cited on pages 9 and 12.)

[do Carmo 1976] Manfredo P. do Carmo. Differential geometry of curves and surfaces. Prentice-Hall, Englewood Cliffs, NJ, 1976. (Cited on page 67.)

[Dodson 1985] Bert Dodson. Keys to drawing. North Light Books, 1985. (Cited on pages 9, 13, 15, 16, 18, 19 and 20.)

[Douglas 1973] David H. Douglas and Thomas K. Peucker. *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*. Cartographica: The International Journal for Geographic Information and Geovisualization, vol. 10, no. 2, pages 112–122, October 1973. (Cited on page 18.)

[Edwards 1979] Betty Edwards. Drawing on the right side of the brain. Tarcher, 1979. (Cited on pages 9, 13, 15 and 18.)

[Eigensatz 2010] Michael Eigensatz, Martin Kilian, Alexander Schiftner, Niloy J Mitra, Helmut Pottmann and Mark Pauly. *Paneling architectural freeform surfaces*. In ACM Transactions on Graphics (TOG), volume 29, page 45. ACM, 2010. (Cited on page 99.)

[Eissen 2008] Koos Eissen and Roselien Steur. Sketching: Drawing techniques for product designers. Bis Publishers, 2008. (Cited on page 82.)

[Eissen 2011] Koos Eissen and Roselien Steur. Sketching: The basics. Bis Publishers, 2011. (Cited on pages 55, 57, 62, 71 and 87.)

[Eitz 2012] Mathias Eitz, James Hays and Marc Alexa. *How do humans sketch objects?* ACM Trans. on Graphics (Proc. SIGGRAPH), vol. 31, no. 4, page 44, 2012. (Cited on pages 9 and 13.)

[Farin 1999] Gerald Farin and Dianne Hansford. *Discrete Coons patches*. Computer Aided Geometric Design, vol. 16, pages 691–700, August 1999. (Cited on page 66.)

[Fernquist 2011] J. Fernquist, T. Grossman and G. Fitzmaurice. *Sketch-sketch revolution: an engaging tutorial system for guided sketching and application learning*. In ACM Symp. on User Interface Software and Technology (UIST), pages 373–382, 2011. (Cited on page 12.)

[Fisher 2007] Matthew Fisher, Peter Schröder, Mathieu Desbrun and Hugues Hoppe. *Design of Tangent Vector Fields*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 26, no. 3, July 2007. (Cited on pages 60 and 65.)

[Flagg 2006] M. Flagg and J.M. Rehg. *Projector-guided painting*. In ACM Symp. on User Interface Software and Technology (UIST), pages 235–244, 2006. (Cited on page 11.)

[Freeman 1991] William T. Freeman and Edward H. Adelson. *The Design and Use of Steerable Filters*. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), vol. 13, pages 891–906, 1991. (Cited on page 60.)

[Fu 2015] Chi-Wing∗ Fu, Peng∗ Song, Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman and Daniel Cohen-Or. *Computational Interlocking Furniture Assembly*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 34, no. 4, pages 091:1–091:11, 2015. ∗ joint first author. (Cited on page 39.)

[Garg 2014] Akash Garg, Andrew O Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly and Max Wardetzky. *Wire mesh design*. ACM Transactions on Graphics (TOG), vol. 33, no. 4, page 66, 2014. (Cited on page 39.)

[Grabler 2009] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva and Takeo Igarashi. *Generating photo manipulation tutorials by demonstration*. ACM Trans. on Graphics (Proc. SIGGRAPH), vol. 28, no. 3, pages 66:1–66:9, 2009. (Cited on page 12.)

[Guennebaud 2010] Gaël Guennebaud, Benoît Jacob*et al.* *Eigen v3*. http://eigen.tuxfamily.org, 2010. (Cited on page 84.)

[Haeberli 1990] Paul Haeberli. *Paint by Numbers: Abstract Image Representations*. SIGGRAPH, vol. 24, no. 4, 1990. (Cited on page 60.)

[Harris 1988] C. Harris and M. Stephens. *A Combined Corner and Edge Detection*. In Proceedings of The Fourth Alvey Vision Conference, 1988. (Cited on page 60.)

[Hertzmann 2000] Aaron Hertzmann and Denis Zorin. *Illustrating smooth surfaces*. In SIGGRAPH, pages 517–526. ACM Press/Addison-Wesley Publishing Co., 2000. (Cited on pages 55, 60, 82 and 86.)

[Hildebrand 2012] Kristian Hildebrand, Bernd Bickel and Marc Alexa. *crdbrd: Shape fabrication by sliding planar slices*. In Computer Graphics Forum, volume 31, pages 583–592. Wiley Online Library, 2012. (Cited on page 39.)

[Hoddinott 2011] B. Hoddinott and J. Combs. Drawing for dummies. 2011. (Cited on pages 9, 15, 16 and 19.)

[Hoddinott 2012] Brenda Hoddinott. *Drawspace, http://www.drawspace.com/*. 2012. (Cited on pages 9 and 15.)

[Igarashi 1999] Takeo Igarashi, Satoshi Matsuoka and Hidehiko Tanaka. *Teddy: A Sketching Interface for 3D Freeform Design*. SIGGRAPH, pages 409–416, 1999. (Cited on page 57.)

[Igarashi 2012] Yuki Igarashi, Takeo Igarashi and Jun Mitani. *Beady: interactive beadwork design and construction*. ACM Transactions on Graphics (TOG), vol. 31, no. 4, page 49, 2012. (Cited on page 38.)

[Instructables 2015] Instructables. `http://www.instructables.com/howto/wire+wrapped+jewelry/`, 2015. Accessed: 2015-05-30. (Cited on page 39.)

[Johnston 2002] Scott F. Johnston. *Lumo: illumination for cel animation*. In Proc. Symp. on Non-Photorealistic Animation and Rendering (NPAR), 2002. (Cited on pages 58, 65 and 91.)

[Joshi 2008] Pushkar Joshi and Nathan A. Carr. *Repoussé: Automatic Inflation of 2D Artwork*. In Proc. of Sketch Based Interfaces and Modeling (SBIM), 2008. (Cited on page 58.)

[Kälberer 2007] Felix Kälberer, Matthias Nieser and Konrad Polthier. *QuadCover - Surface Parameterization using Branched Coverings*. Computer Graphics Forum, vol. 26, no. 3, pages 375–384, September 2007. (Cited on page 69.)

[Kang 2007] Henry Kang, Seungyong Lee and Charles K Chui. *Coherent line drawing*. In ACM Symp. on Non-photorealistic Animation and Rendering (NPAR), pages 43–50, 2007. (Cited on page 60.)

[Kang 2009] Henry Kang, Seungyong Lee and Charles K Chui. *Flow-based image abstraction*. IEEE Trans. on Visualization and Computer Graphics (TVCG), vol. 15, no. 1, pages 62–76, 2009. (Cited on page 60.)

[Kaplan 2005] Craig S. Kaplan and Robert Bosch. *TSP Art*. Bridges: Mathematical Connections in Art, Music and Science, pages 301–308, 2005. (Cited on page 39.)

[Kass 1987] Michael Kass and Andrew Witkin. *Analyzing oriented patterns*. Computer vision, graphics, and image processing, vol. 37, no. 3, pages 362–385, 1987. (Cited on page 60.)

[Kirkpatrick 1983] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi*et al.* *Optimization by simmulated annealing*. science, vol. 220, no. 4598, pages 671–680, 1983. (Cited on page 43.)

[Knill 1992] David C. Knill. *Perception of surface contours and surface shape: from computation to psychophysics*. Journal of Optical Society of America, vol. 9, no. 9, pages 1449–1464, 1992. (Cited on page 69.)

[Knöppel 2013] Felix Knöppel, Keenan Crane, Ulrich Pinkall and Peter Schröder. *Globally optimal direction fields*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 32, no. 4, page 59, 2013. (Cited on pages 56, 60 and 65.)

[Kohr. 2012] Matt Kohr. *Ctrl+Paint, http://www.ctrlpaint.com/*. 2012. (Cited on pages 9, 15, 16 and 17.)

[Kolmogorov 2006] Vladimir Kolmogorov. *Convergent Tree-Reweighted Message Passing for Energy Minimization*. IEEE Trans. Pattern Analysis Machine Intelligence (PAMI), vol. 28, no. 10, pages 1568–1583, October 2006. (Cited on pages 72 and 94.)

[Kyprianidis 2011] Jan Eric Kyprianidis and Heenry Kang. *Image and Video Abstraction by Coherence-Enhancing Filtering*. Computer Graphics Forum, vol. 30, no. 2, pages 593–602, 2011. (Cited on pages 60 and 64.)

[Laviole 2012] Jeremy Laviole and Martin Hachet. *PapARt: interactive 3D graphics and multi-touch augmented paper for artistic creaton*. In 3DUI - IEEE Virtual Reality Conference, 2012. (Cited on page 11.)

[Lee 2011] Y.J. Lee, C.L. Zitnick and M.F. Cohen. *ShadowDraw: real-time user guidance for freehand drawing*. ACM Trans. on Graphics (Proc. SIGGRAPH), vol. 30, no. 4, page 27, 2011. (Cited on pages 9 and 11.)

[Lefebvre 2006] Sylvain Lefebvre and Hugues Hoppe. *Appearance-space Texture Synthesis*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 25, no. 3, 2006. (Cited on pages 55 and 60.)

[Li 2011] Xian-Ying Li, Tao Ju, Yan Gu and Shi-Min Hu. *A Geometric Study of V-style Pop-ups: Theories and Algorithms*. ACM Transactions on Graphics (Proc. of SIGGRAPH), vol. 30, no. 4, pages 98:1–98:10, July 2011. (Cited on page 38.)

[Limpaecher 2013] Alex Limpaecher, Nicolas Feltman and Michael Cohen. *Real-Time Drawing Assistance Through Crowdsourcing*. ACM Trans. on Graphics (Proc. SIGGRAPH), 2013. (Cited on page 11.)

[Liu 2011] Yang Liu, Weiwei Xu, Jun Wang, Lifeng Zhu, Baining Guo, Falai Chen and Guoping Wang. *General planar quadrilateral mesh design using conjugate direction field*. ACM Transactions on Graphics (Proc. SIGGRAPH Asia), vol. 30, no. 6, page 140, 2011. (Cited on pages 60 and 82.)

[Lopez-Moreno 2013] Jorge Lopez-Moreno, Stefan Popov, Adrien Bousseau, Maneesh Agrawala and George Drettakis. *Depicting Stylized Materials with Vector Shade Trees*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 32, no. 4, 2013. (Cited on page 58.)

[Mamassian 1998] Pascal Mamassian and Michael S. Landy. *Observer biases in the 3D interpretation of line drawings*. Vision research, vol. 38, no. 18, pages 2817–2832, 1998. (Cited on page 69.)

[Martin 2001] David Martin, Charless Fowlkes, Doron Tal and Jitendra Malik. *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*. In IEEE International Conference on Computer Vision, pages 416–423, 2001. (Cited on page 18.)

[McIntosh 2007] Jim McIntosh. Wire wrapping: The basics and beyond. CreateSpace Independent Publishing Platform, 2007. (Cited on page 39.)

[Mori 2007] Yuki Mori and Takeo Igarashi. *Plushie: an interactive design system for plush toys*. In ACM Transactions on Graphics (TOG), volume 26, page 45. ACM, 2007. (Cited on page 38.)

[Nealen 2007] Andrew Nealen, Takeo Igarashi, Olga Sorkine and Marc Alexa. *FiberMesh: Designing Freeform Surfaces with 3D Curves*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 26, no. 3, July 2007. (Cited on page 57.)

[Nehab 2005] Diego Nehab, Szymon Rusinkiewicz, James Davis and Ravi Ramamoorthi. *Efficiently Combining Positions and Normals for Precise 3D Geometry*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 24, no. 3, August 2005. (Cited on page 97.)

[Nicolaides 1969] K. Nicolaides. The natural way to draw: A working plan for art study. Houghton Mifflin Co., 1969. (Cited on pages 9 and 13.)

[Noris 2013] Gioacchino Noris, Alexander Hornung, Robert W Sumner, Maryann Simmons and Markus Gross. *Topology-driven vectorization of clean line drawings*. ACM Transactions on Graphics, vol. 32, no. 1, page 4, 2013. (Cited on page 59.)

[Orbay 2011] Günay Orbay and Levent Burak Kara. *Beautification of design sketches using trainable stroke clustering and curve fitting*. IEEE Trans. on Visualization and Computer Graphics (TVCG), vol. 17, no. 5, pages 694–708, 2011. (Cited on page 59.)

[Panozzo 2014] Daniele Panozzo, Enrico Puppo, Marco Tarini and Olga Sorkine-Hornung. *Frame Fields: Anisotropic and Non-Orthogonal Cross Fields*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 33, no. 4, 2014. (Cited on page 61.)

[Pipes 2007] Alan Pipes. Drawing for designers. Laurence King, 2007. (Cited on pages 56 and 57.)

[Praun 2000] Emil Praun, Adam Finkelstein and Hugues Hoppe. *Lapped Textures*. SIGGRAPH, 2000. (Cited on page 60.)

[Qu 2006] Yingge Qu, Tien-Tsin Wong and Pheng-Ann Heng. *Manga Colorization*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 25, no. 3, July 2006. (Cited on page 57.)

[Ray 2006] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer and Pierre Alliez. *Periodic global parameterization*. ACM Transactions on Graphics, vol. 25, no. 4, pages 1460–1485, 2006. (Cited on pages 55, 60, 70 and 75.)

[Ray 2008] Nicolas Ray, Bruno Vallet, Wan Chiu Li and Bruno Lévy. *N-symmetry Direction Field Design*. ACM Transactions on Graphics, vol. 27, no. 2, 2008. (Cited on pages 60 and 76.)

[Ray 2009] Nicolas Ray, Bruno Vallet, Laurent Alonso and Bruno Levy. *Geometry-aware direction field processing*. ACM Transactions on Graphics, vol. 29, no. 1, page 1, 2009. (Cited on page 56.)

[Rivers 2012] Alec Rivers, Andrew Adams and Frédo Durand. *Sculpting by numbers*. ACM Trans. on Graphics (Proc. SIGGRAPH Asia), vol. 31, no. 6, 2012. (Cited on page 11.)

[Rother 2004] Carsten Rother, Vladimir Kolmogorov and Andrew Blake. *"GrabCut": interactive foreground extraction using iterated graph cuts*. ACM Trans. on Graphics (Proc. SIGGRAPH), vol. 23, no. 3, 2004. (Cited on page 16.)

[Schmidt 2009a] Ryan Schmidt, Azam Khan, Gord Kurtenbach and Karan Singh. *On Expert Performance in 3D Curve-Drawing Tasks*. In Proc. Symp. Sketch-Based Interfaces and Modeling (SBIM), 2009. (Cited on page 97.)

[Schmidt 2009b] Ryan Schmidt, Azam Khan, Karan Singh and Gord Kurtenbach. *Analytic Drawing of 3D Scaffolds*. ACM Trans. on Graphics (Proc. SIGGRAPH Asia), vol. 28, no. 5, 2009. (Cited on page 12.)

[Schmidt 2013] R. Schmidt and M. Ratto. *Design-to-Fabricate: Maker Hardware Requires Maker Software*. Computer Graphics and Applications, IEEE, vol. 33, no. 6, pages 26–34, Nov 2013. (Cited on page 38.)

[Shao 2012] Cloud Shao, Adrien Bousseau, Alla Sheffer and Karan Singh. *CrossShade: shading concept sketches using cross-section curves*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 31, no. 4, page 45, 2012. (Cited on pages 56, 58, 59, 65, 66, 67, 69, 71, 72, 87, 91 and 92.)

[Shi 1994] J. Shi and C. Tomasi. *Good features to track*. In IEEE Conference on Computer Vision and Pattern Recognition, 1994. (Cited on pages 19 and 23.)

[Skouras 2012] Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel and Markus Gross. *Computational Design of Rubber Balloons*. Computer Graphics Forum (proc. Eurographics), vol. 31, no. 2, pages 835–844, May 2012. (Cited on page 38.)

[Skouras 2014] Mélina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun and Markus Gross. *Designing Inflatable Structures*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 33, no. 4, pages 63:1–63:10, 2014. (Cited on page 38.)

[Soga 2009] Masato Soga, Shota Kuriyama and Hirokazu Taki. *Sketch Learning Environment with Diagnosis and Drawing Guidance from Rough Form to Detailed Contour Form*. T. Edutainment, vol. 3, pages 129–140, 2009. (Cited on page 12.)

[Soille 2003] Pierre Soille. Morphological image analysis: Principles and applications. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2 édition, 2003. (Cited on page 40.)

[Stevens 1981] Kent A Stevens. *The visual interpretation of surface contours*. Artificial Intelligence, vol. 17, no. 1, pages 47–73, 1981. (Cited on pages 62, 66 and 69.)

[Sýkora 2009] Daniel Sýkora, John Dingliana and Steven Collins. *LazyBrush: Flexible Painting Tool for Hand-drawn Cartoons*. Computer Graphics Forum (Proc. EUROGRAPHICS), vol. 28, no. 2, 2009. (Cited on page 57.)

[Sýkora 2011] Daniel Sýkora, Mirela Ben-Chen, Martin Čadík, Brian Whited and Maryann Simmons. *TexToons: Practical Texture Mapping for Hand-drawn Cartoon Animations*. In Proc. Symp. on Non-photorealistic Animation and Rendering (NPAR), pages 75–83, 2011. (Cited on page 58.)

[Sýkora 2014] Daniel Sýkora, Ladislav Kavan, Martin Čadík, Ondřej Jamriška, Alec Jacobson, Brian Whited, Maryann Simmons and Olga Sorkine-Hornung. *Ink-and-Ray: Bas-Relief Meshes for Adding Global Illumination Effects to Hand-Drawn Characters*. ACM Transaction on Graphics, vol. 33, 2014. (Cited on page 58.)

[Tanenbaum 2013] Joshua G. Tanenbaum, Amanda M. Williams, Audrey Desjardins and Karen Tanenbaum. *Democratizing Technology: Pleasure, Utility and Expressiveness in DIY and Maker Practice*. In Proc. of the SIGCHI Conference on Human Factors in Computing Systems, pages 2603–2612. ACM, 2013. (Cited on page 38.)

[Umetani 2012] Nobuyuki Umetani, Takeo Igarashi and Niloy J. Mitra. *Guided Exploration of Physically Valid Shapes for Furniture Design*. ACM Transactions on Graphics (Proc. of SIGGRAPH), vol. 31, no. 4, 2012. (Cited on page 38.)

[Umetani 2014] Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt and Takeo Igarashi. *Pteromys: Interactive Design and Optimization of Free-formed Free-flight Model Airplanes*. ACM Transactions on Graphics (Proc. of SIGGRAPH), vol. 33, no. 4, pages 65:1–65:10, July 2014. (Cited on page 38.)

[Vergne 2012] Romain Vergne, Pascal Barla, Roland W. Fleming and Xavier Granier. *Surface Flows for Image-based Shading Design*. ACM Transactions on Graphics (Proc. SIGGRAPH), vol. 31, no. 4, pages 94:1–94:9, July 2012. (Cited on page 58.)

[Wächter 2006] Andreas Wächter and Lorenz T. Biegler. *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*. Math. Program., vol. 106, pages 25–57, May 2006. (Cited on page 84.)

[Weickert 1999] Joachim Weickert. *Coherence-Enhancing Diffusion Filtering*. International Journal of Computer Vision, vol. 31, no. 2-3, 1999. (Cited on page 60.)

[WigJig 2015] WigJig. www.wigjig.com, 2015. Accessed: 2015-05-30. (Cited on pages 36, 39 and 47.)

[Winnemöller 2009] Holger Winnemöller, Alexandrina Orzan, Laurence Boissieux and Joëlle Thollot. *Texture Design and Draping in 2D Images*. Computer Graphics Forum (Proc. Symp. on Rendering), vol. 28, no. 4, 2009. (Cited on pages 58 and 83.)

[Wong 2011] Fernando J. Wong and Shigeo Takahashi. *A Graph-based Approach to Continuous Line Illustrations with Variable Levels of Detail*. Computer Graphics Forum, vol. 30, no. 7, pages 1931–1939, 2011. (Cited on page 39.)

[Xu 2014] Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae and Karan Singh. *True2Form: 3D Curve Networks from 2D Sketches via Selective Regularization*. Transactions on Graphics (Proc. SIGGRAPH), vol. 33, no. 4, 2014. (Cited on pages 58 and 97.)

[Yang 2011] Yong-Liang Yang, Yi-Jun Yang, Helmut Pottmann and Niloy J Mitra. *Shape space exploration of constrained meshes*. ACM Trans. Graph., vol. 30, no. 6, page 124, 2011. (Cited on page 99.)

[Zhuang 2013] Yixin Zhuang, Ming Zou, Nathan Carr and Tao Ju. *A general and efficient method for finding cycles in 3D curve networks*. ACM Transactions on Graphics (Proc. SIGGRAPH Asia), vol. 32, no. 6, 2013. (Cited on page 69.)

Computer drawing tools for assisting learners, hobbyists and professionals

Abstract: Drawing is the earliest form of visual depiction. The goal of this thesis is to facilitate and accelerate drawing for amateurs as well as for expert designers and illustrators, employing computer graphics, image processing and interaction techniques. As this is a broad spectrum to tackle, we identify three specific problems related to drawing and propose computer tools to help users overcome the main challenges on each domain.

In Chapter 2 we present an interactive drawing tool to help beginners practice drawing-by-observation techniques. We build on a number of traditional techniques to help people gain consciousness of the shapes in a scene. We automatically extract visual guides from a model photograph and offer corrective feedback to guide their reproduction in the drawing.

In Chapter 3 we propose a tool that helps users create wire wrapped jewelry. This technique of handmade jewelry can be seen as a form of line drawing with metal wires. The presented method assist the user in the main challenges of creating 2D wire-wrapped jewelry from a drawing: decomposing the input into a set of wires, and bending the wires to give them shape.

In Chapter 4 we propose a method to help designers enrich their drawings with color and shading. Professionals frequently draw curvature lines to convey bending of smooth surfaces in concept sketches. We exploit this information and extrapolate curvature lines in a rough concept sketch. This extrapolation allows us to recover the intended 3D curvature and surface normal at each pixel, which we use to compute shading and texturing over the sketch.

Outils de dessin informatique pour les débutants, les passionnés et les professionnels

Résumé: Le dessin est la plus ancienne forme de représentation visuelle. Le but de cette thèse est de faciliter et d'accélérer le dessin pour les amateurs ainsi que pour les dessinateurs experts en utilisant des techniques de traitement d'image et d'interaction. Comme ce but couvre un large spectre d'applications, nous identifions trois problèmes spécifiques liés au dessin, et proposons des outils pour aider les utilisateurs à surmonter les principaux défis sur chaque domaine.

Dans le chapitre 2, nous présentons un outil de dessin interactif pour aider des débutants à pratiquer les techniques traditionnelles de dessin par observation. Nous construisons cet outil autour de techniques traditionelle pour aider les gens à acquérir la conscience des formes dans une scène. Nous extrayons automatiquement des guides visuels à partir d'une photographie. L'interface de l'outil affiche ces informations et offre un retour pour guider l'utilisateur dans la reproduction du dessin.

Dans le chapitre 3, nous proposons un outil qui permet aux utilisateurs de créer des bijoux par pliage de fils de fer. Cette forme de bijoux faits à la main peut être considérée comme une forme de dessin à base de fil de fer. La méthode présentée aide l'utilisateur dans les principaux défis de la création de bijoux à partir d'un dessin: la décomposition de l'entrée dans un ensemble de fils, et le pliage des fils pour leur donner forme.

Dans le chapitre 4, nous proposons une méthode pour aider les designers à enrichir leurs dessins avec de la couleur et de l'ombrage. Les designers tracent souvent des lignes de courbure pour représenter la forme des surfaces lisses dans des esquisses. Nous exploitons cette information et extrapolons les lignes de courbure dans le design. Cette extrapolation nous permet d'estimer la courbure 3D du dessin et la normale de la surface en chaque pixel, pour créer des ombres et des textures sur l'esquisse.