Diss. ETH No. 25549

Computational fabrication of 3D shapes: Enabling makers through novel geometric algorithms

A thesis submitted to attain the degree of
Doctor of Sciences of **ETH Zurich**
(Dr. sc. ETH Zurich)

presented by
Christian Eduard Schüller
MSc ETH CS, ETH Zurich, Switzerland
born on 13.05.1984
citizen of Switzerland

accepted on the recommendation of
Prof. Dr. Olga Sorkine-Hornung, examiner
Prof. Dr. Marc Alexa, co-examiner
Prof. Dr. Daniele Panozzo, co-examiner
Prof. Dr. Stelian Coros, co-examiner

2018

Abstract

We are witnessing a new revolution in the area of fabrication and manufacturing. The most recent generation of digital fabrication devices like 3D printers, laser cutters and 4-axis desktop milling machines, make it considerably easier for non-professionals to fabricate their own custom designed tools and objects at a reasonable price. This evolution will eventually liberate many from having to buy pre-built products and already allows a high level of individualized object design and customization. Over the past few years, an active community of *makers* has evolved, that constantly pushes the development of digital fabrication devices in the direction of smaller, more affordable tabletop and lab-sized machines. With this, there comes the need for user-friendly software and methods to create and work with their input data, since many of the devices' users do not have any special technical knowledge. Depending on the fabrication method, a designed shape needs to satisfy certain constraints and be physically feasible, especially if one wishes to replicate or approximate highly complex digital models. In this thesis, we explore and develop novel computational design methods that extend the capabilities of traditional crafting and manufacturing techniques to create 3D shapes. Specifically, we are interested in identifying instances of design processes that can be simplified and improved algorithmically and made readily accessible to the maker community through the use of digital fabrication techniques. We show how these approaches allow the creation of objects with a new level of quality and complexity, and think that this work will enable novel applications not just for the maker community but also potentially for industrial manufacturing.

Zusammenfassung

Wir sind Zeugen einer neuen Revolution im Bereich der Fabrikation und Fertigung. Die neuste Generation von bürotauglichen, digitalen Fabrikationsgeräten, wie zum Beispiel 3D Drucker, Laserschneidegeräte oder kompakten CNC-Fräsmaschinen, ermöglicht auch privaten Benutzern die Herstellung von selbst entworfenen Werkzeugen und Objekten zu erschwinglichen Preisen. Diese Entwicklung wird einmal den Einzelnen vom Kauf vorgefertigter Produkte befreien und ermöglicht stattdessen einen hohen Grad an individueller Gestaltung von massgeschneiderten Lösungen. Seit einigen Jahren entwickelt sich eine aktive Gemeinschaft von so genannten *Makers*, die die Entwicklung von diesen kleineren, einfach zu benutzenden, digitalen Fabrikationsgeräten mit viel Pioniergeist vorantreiben. Allerdings bedarf es für deren erfolgreiche Anwendung auch neue digitale Bearbeitungsmethoden, welche fähig sind, die entsprechend Eingabedaten zu erstellen und zu editieren. Viele der Benutzer haben kein spezielles technisches Vorwissen und sind deshalb auf einfache und bedienerfreundliche Softwarelösungen angewiesen. Oft müssen im Kontext der Herstellungen von physischen Objekten mit unterschiedlichen Fabrikationsmethoden, gewisse verfahrenstechnische Bedingungen und Einschränkungen eingehalten werden, besonders wenn komplexe digitale Modelle repliziert oder deren Form angenähert werden sollen. In dieser Arbeit entwickeln wir neue computergestützte Methoden und Algorithmen, die dies erleichtern und den Entwurf und die Fabrikation solcher dreidimensionaler Objekte ohne grosse Mühe ermöglichen. Wir fokussieren uns dabei auf traditionell handwerkliche und industrielle Verfahren, um diese der Maker-Gemeinschaft besser zugänglich zu machen. Wir zeigen anhand dieser Beispiele, wie neue computergestützte Methoden und Algorithmen den Entwurf und die Fabrikation von Objekten von höherer Qualität und Komplexität ermöglichen und denken, dass die gezeigten Ansätze zu neuen Anwendungen, nicht nur in der Maker-Gemeinschafft, sondern auch in der Industrie führen können.

Acknowledgments

I am deeply grateful to Olga Sorkine-Hornung for the chance I was given to pursue my Ph.D. at the Interactive Geometry Lab. I would like to thank her for the continuous support and guidance over all these years and the trust and patience I received during the course of my research. She always had an open ear in times of challenge and encouraged me to explore my own approaches and ideas.

Daniele Panozzo, mentor, friend and closest collaborator, advised and thought me countless things that have laid the foundations of my work in many hours of fruitful discussions. His positive and constructive way of thinking is inspiring and working with him is an experience I do not want to miss. I also owe a large amount of gratitude to all my coauthors, Anselm Grundhöfer, Henning Zimmer, Evgeni Sorkine and in particular to Roi Poranne, who advised and helped me during the last part of my Ph.D. with his profound mathematical knowledge and critical thinking.

Furthermore, I would like to thank everybody that lent a hand in my projects when I needed one: Emily Whiting, Kaan Yücer, Vaclav Hnizda, Brian McWilliams and Renana Poranne for their help with the paper submission videos, as well as Alessia Marra, Jan Wezel, Isabelle von Salis and the tailors from jaelsigner.ch and atelier-renaissance.ch for the support and the creation of the digital/physical models and the Mirko Meboldt and the staff of the Raplab D-Arch for the free access to their fabrication devices. I am grateful for the valuable discussions with Bernhard Thomaszewski, Olga Diamanti, Peter Kaufmann and Wenzel Jakob, Oliver Glauser and Michael Rabinovich. A special thank goes to Olga Diamanti for proof reading my thesis and Marianna Berger for all the administrative work.

I have been very lucky to work at ETH Zurich, with so many inspiring and open-minded people from all over the world. I would like to thank my friends and colleagues from CGL, IGL, CVG and DRZ for all the great moments and discussions we had.

I am deeply thankful to my friends and family. You stayed with me through all ups and downs during this exciting time. Nothing would have been possible without the unconditionally support and love from the woman of my life, Linda Hanhart. Thank you for standing by me!

This work was supported in part by the ERC grant iModel (StG-2012-306877).

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

With the newest generation of digital fabrication devices like 3D printers, laser cutters and 4-axis desktop milling machines, we are about to experience a new revolution in the area of fabrication. It is part of what some call the "Third Industrial Revolution" [Rif13] and might eventually lead to the democratization of manufacturing, where production will happen in a highly individualized and distributed fabrication infrastructure. Customized parts and products are expected to be produced collectively by many different suppliers, professionals but especially private individuals. First attempts already exist with initiatives like Additively [Add18], which is a network of independent digital fabrication device owners who offer to share them as a service. Ultimately, this will empower people to create their own customized machines and parts at low cost, achieving unlimited creative freedom.

At the core of this vision lies affordable access to digital fabrication devices. But equally importantly, there is the need for software and methods to create and work with the user-provided input data. Since the mid 1960s, computer aided design (CAD) has been a major driving force for research in computational geometry, computer graphics, and discrete differential geometry [PBCW07]. These disciplines have created the basis for computer-aided design technologies to create, modify, simulate, optimize and visualize the shape of tools and machinery parts, as well as furniture, appliances or even jewelery. In the beginning, CAD software often required deep technical expertise and was used mainly by specialists and professionals. More recently, with the development of affordable, compact and easier to use devices like the newest 3D tabletop printers, a thriving community of makers and artists

has evolved, who started to explore the new possibilities of these tools. At the same time, the development of free and user-friendly modeling tools like Blender [Com18] or SketchUp [Ske18] enabled these makers to exploit their full potential. The maker community actively pushes this development further. So-called fab labs [Fou], small-scaled shared workshops, offer access to digital fabrication to everybody and are becoming increasingly more commonplace. There is a variety of other closely related initiatives that share the same philosophy as well as technology, like the DIY movement, open source hardware, maker culture, and the free and open source movement.

Currently, in the industry, digital fabrication is employed in all phases of manufacturing, ranging from rapid prototyping to the fabrication of end-user products. Although the maker community's devices do not yet have the same working dimensions and operational speed, they are often good enough to create parts of comparable quality, at the cost of a limited number of available materials and the need of manual fine-tuning. Besides purely functional parts for machines and other mechanical components, the physical manufacturing of 3D shapes from digital models is of great importance in many other fields. In industrial/product design and architecture, tangible prototypes help to better visualize and convey the look and feel of the envisioned product or building and drastically reduces the development time. Authentic replicas of historical artifacts allow museums to exhibit them to a broader audience in a unprotected environment. In fashion and interior/artistic design, digitally designed, complexly shaped objects and surfaces can be brought alive and exhibited.

Even though there is an abundance of readily available 3D shapes on the Internet, they usually cannot be directly used without further modification. Depending on the fabrication method, a designed shape needs to satisfy certain fabrication-related constraints and be physically feasible. For example, 3D printing extrusion technology often has a limitation on the angle for overhanging parts if no support structures or materials are used. CNC milling, on the other hand, depends on complicated tool path planning, which needs to consider the type of material used and usually cannot handle arbitrary geometry. In the case of laser cutting, one is limited to work with flat materials and the final 3D shape needs to be assembled from multiple pieces. Therefore, the initial target shape can often only be approximated, given the different requirements and constraints. All of this adds a layer of complexity to the design process which remains one of the biggest challenges for many people in the maker community. Without the help of smart computational tools this is almost impossible to get right, even for professionals. Therefore, the exploration of new methods and algorithms that enable the physical

rendition of user-imagined 3D models of arbitrary complexity is subject to active research in Computational Design and Fabrication.

1.1 Topics in this thesis

In this thesis, we explore and develop novel computational design methods that extend the capabilities of traditional crafting and manufacturing techniques to create 3D shapes. Specifically, we are interested in identifying instances of design processes that can be simplified and improved with algorithms and made readily accessible to the maker community through the use of digital fabrication techniques. We show how these approaches allow the creation of objects with a new level of quality and complexity, and believe that this work will enable novel applications not just for the maker community but also potentially for industry.

In the context of this thesis, we developed tailored algorithms to address three problem-specific challenges found in various methods of fabrication. Even though each proposed approach was designed with a specific fabrication method in mind, we think that the developed mathematical methods and models are of broader use and can inspire new solutions for other types of applications. Because of the diverse properties and constraints involved in the various fabrication methods, it would make little sense to develop a single unified framework. Instead, we chose to independently demonstrate the power and flexibility of the different problem specific approaches.

Relief creation from digital 3D models. The art of creating reliefs has fascinated humankind for millennia and it is extensively used on many walls of buildings, coins, medals, pottery and other surfaces. Traditionally, reliefs are made by chiselling or carving away material from a solid block of material, such that the remaining parts arouse the impression of a 3D scene raised above the background plane, within a thin space. This craft requires a lot of artistic skills and experience. Depending on the material, the creation of a single relief can take up to weeks of hard work. Nowadays, with the help of modern CNC milling machines, the fabrication can be done in a few hours. But the initial creation of the digital relief model remains a challenge. We therefore developed an unified framework to automatically deform existing 3D models, such that they are confined to a given limited space, while optimally approximating the original visual appearance, if observed from a designated viewpoint. This method allows makers to easily compute and

fabricate their own reliefs from existing 3D models, not only on flat pieces but also on surfaces that curve or even are split into multiple pieces.

Replication of 3D colored objects with thermoforming. In the industry, thermoforming is the dominant mass production technique for the fabrication of plastic objects like disposable cups, containers, lids and other products for the retail industries. In this process a plastic sheet is heated to a pliable forming temperature and deformed to a given mold. The plastic keeps its shape after cooling and the final object only needs to be trimmed from it. Thermoforming is also used for prototyping or the fast replication of 3D shapes from existing objects. Smaller tabletop or lab size vacuum forming machines can be found in many workshops or fab lab spaces. Unfortunately, the automatic coloring or image texturing of these thermoformed surfaces is a complicated post-process, requiring special tools, which are usually not accessible to the maker community. Therefore, we propose a method to compute and print a pre-distorted image onto the flat plastic sheet, that compensates for the shape-specific distortion and perfectly aligns with geometric features after thermoforming. This makes it possible to faithfully reproduce colored digital 3D models with affordable off-the-shelf tools.

Design and fabrication of zippable 3D shapes. Handicrafts that involve fabric materials, pose another interesting design case to create 3D shapes. Tailoring and dressmaking is still one of the most manual labor-intensive industries today. Additionally, many individuals are interested in creating their own dresses or bags themselves. The laser technology has brought a lot of automation and speedup in the pattern cutting process, but sewing the different parts together still mostly needs to be done by hand and can be very time consuming. In particular, assembling and stuffing shells of 3D objects like pillows or plush toys can be extremely challenging for people with little experience. Inspired by the concept of so-called zip-it bags [zip17], we developed a novel design and fabrication method to create zippable 3D shapes from fabric. Our algorithm interactively supports the user to compute a single flat ribbon-like piece of fabric that, once a zipper is attached to its boundary, can be simply zipped up to assemble the original target shape. While these *Zippables* are a lot of fun to explore and play with, we see another interesting applications of this technique: pipe cladding could benefit from precut long strips of insulation material that quickly wraps around pipes, reducing the assembly time needed. This is an example of a possible direct extension of our proposed problem-specific approaches to a different domain of industrial manufacturing.

1.2 Contributions of this thesis

Below we outline the major contributions of the work presented in this thesis.

- An algorithm to deform a given 3D shape, such that it fits within a given limited space and resembles the visual appearance of the original model, if viewed from a designated viewpoint. This is achieved by solving a novel convex optimization problem that minimizes the perceived shading difference between the deformed and original model surface assuming a Lambertian material model. The algorithm allows to construct anamorphic relief sculptures on non planar, possibly split backplanes.
- An algorithm to simulate a simplified visco-plastic material model of a plastic sheet in the vacuuming forming process. The discrete model only has four independent, free parameters which are automatically determined by a grid search, based on a single calibration sample. The result of the simulation is used to compute the pre-distorted image that needs to be printed on the plastic sheet, such that it aligns with the texture of the original target model.
- Description of a simple of-the-shelf setup to transfer a color image onto a plastic sheet and instructions on how to create a heat resistant thermoforming mold with a simple PLA 3D printer.
- A computational method to create the cutting plans for a single flat piece of fabric, that can be quickly assembled to a 3D target model if zipped up along the boundary. The algorithm includes: A step to create a single spiraling curve over the entire surface of a 3D model, given a cylindrical segmentation; A remeshing of the surface after being cut along the curve to create a developable surface that can be flattened without distortion; A computation of a 3D offset to incorporate the zipper's dimension.
- A novel approach for the construction of a fixation rig that helps to align and attach zipper tapes to flat fabric with curved boundaries .

1.3 Thesis outline

The dissertation is divided into six chapters. The remaining chapters are organized as follows.

Chapter 2 presents work relating to computational design and fabrication of 3D shapes. In a structured manner, we introduce the research that inspired our work as well as follow-up research.

Chapter 3 presents our approach of digital relief creation. We first define the related deformation problem and derive a convex optimization to find a unique solution. We showcase the effectiveness of our approach on a variety of highly detailed and complex fabricated 3D models.

Chapter 4 presents an approach to color the surface of thermoformed plastic objects. We first describe a simplified material model that can be used to simulate a thin plastic sheet in the thermoforming process. We then show how to find the 2D image that needs to be printed on the flat plastic sheet to replicate the original texture of the target object. We also propose a simple off-the-shelf setup to create a heat resistant mold and an approach to print on plastic sheets, that allows makers to use this process without having access to expensive machines.

Chapter 5 presents our approach to create so called *Zippables*: Flat pieces of fabric with a zipper attached around their boundaries, that can be quickly zipped up to assemble 3D target shapes. We first show how to create a regular spiraling curve on the surface of a 3D model, which then serves as the cutting line to generate a single ribbon-like piece. After a special remeshing the surface will be developable and can be flattened without distortion. Finally, we introduce the construction of a fabrication rig that helps to align and attach the zipper tape to flat fabric with curved boundaries.

Chapter 6 summarizes our contributions and reflects on potential avenues for future work.

Appendix A provides the derivations to turn the quadratic programming problem in Chapter 3 into a conic optimization.

Appendix B provides the details on how to reconstruct the vertex positions of a triangle, given its edge lengths.

Appendix C provides the details about the zipper-curve offsetting and proofs that it results in two new curves of equal length.

1.4 Publications

This thesis resulted in the following peer-reviewed publications:

[tPSH14] Christian Schüller, Daniele Panozzo, and Olga Sorkine-Hornung. Appearance-mimicking surfaces. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, 33(6):216:1–216:10, 2014

[tPG$^+$16] Christian Schüller, Daniele Panozzo, Anselm Grundhöfer, Henning Zimmer, Evgeni Sorkine, and Olga Sorkine-Hornung. Computational thermoforming. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 35(4), 2016

[tPSH18] Christian Schüller, Roi Poranne, and Olga Sorkine-Hornung. Shape representation by zippables. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 37(4), 2018

During the course of this thesis, the following peer-reviewed papers were also published:

[tKPSH13] Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. Locally injective mappings. *Computer Graphics Forum (proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing)*, 32(5):125–135, 2013

[SJP$^+$13] Leonardo Sacht, Alec Jacobson, Daniele Panozzo, **Christian Schüller**, and Olga Sorkine-Hornung. Consistent volumetric discretizations inside self-intersecting surfaces. *Computer Graphics Forum (proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing)*, 32(5):147–156, 2013

[PtP$^+$14] Kazim Pal, **Christian Schüller**, Daniele Panozzo, Olga Sorkine-Hornung, and Tim Weyrich. Content-aware surface parameterization for interactive restoration of historical documents. *To appear in Computer Graphics Forum (Proc. Eurographics)*, 33(2), 2014

[PAB+16] Kazim Pal, Nicole Avery, Pete Boston, Alberto Campagnolo, Caroline De Stefani, Helen Matheson-Pollock, Daniele Panozzo, Matthew Payne, **Schüller, Christian**, Chris Sanderson, Chris Scott, Philippa Smith, Rachael Smither, Olga Sorkine-Hornung, Ann Stewart, Emma Stewart, Patricia Stewart, Melissa Terras, Bernadette Walsh, Laurence Ward, Liz Yamada, and Tim Weyrich. Digitally reconstructing the great parchment book: 3D recovery of fire-damaged historical documents. *Digital Scholarship in the Humanities*, December 2016

CHAPTER 2

Related Work

The main focus of this dissertation lies on the computer-assisted design and fabrication of 3D shapes, based on traditional crafting and manufacturing techniques. While in recent years additive manufacturing methods have become very popular and enabled new ways to create highly detailed shapes from different materials, many traditional crafting and manufacturing techniques remain of high importance and have been continuously evolving in modern times. They also have greatly benefited from the research and progress in the field of digital fabrication, especially in the context of making them more accessible to the maker community. In this chapter, we provide an overview of the most recent literature related to these developments. We summarize computational design methods which enabled and advanced the fabrication of 3D shapes from digital models and categorize them in terms of the associated crafting or manufacturing technique. We believe that this gives an informative overview of the different processes available to makers and might even help discover interesting new problems not yet tackled by current research.

Two recent state-of-the-art reports discuss a common set of works from different perspectives. Bickel et al. [BCMP18] summarize current research about *stylized* fabrication methods in terms of different phenomena that are taken into account when generating the final physical representation: shape, material, lighting and shadow, decomposition and 'printing the unprintable'. Bermano et al. [BFR17] review computational fabrication approaches according to the designer's higher-level goals like structural integrity, deformation and motion, appearance and special function. They both include additive

manufacturing methods and also discuss approaches that use light and shadows.

2.1 Sculpting and carving

Sculpting is one of the most ancient techniques to create 3D objects from almost any solid material. It has been used throughout all centuries dating back to the earliest societies capable of some form of stone work. Sculptures are either made in the round, such as statues, or are partly attached to a background surface and refereed to as reliefs. Among many other purposes, they have been used as lasting depictions of religious and political events and portraits.

Not surprisingly, digital sculpting has become an essential tool in the creation of computer graphics artwork, industrial design and many other 3D modeling applications. Professional tools like *ZBrush*, *Mudbox*, *Blender* and others offer similar techniques to manipulate a digital model as if it were made from a real-life material such as clay. There is a great deal of literature on this topic which we will not cover here, but instead provide some good starting points [MQW01, CCP+05, PFGL08]. On the fabrication side, there is the modern 5-axis CNC-milling technology [LXG10], capable of creating free-form surfaces from almost any material. While abundant work exists on this topic as well, we will focus on novel approaches which are more accessible for the maker community.

[RAD12] uses a conventional projector-camera pair to guide an unskilled user in the manual sculpting process to replicate a digital model from a plastic material by visualizing deviations from the target surface with projected colors. Similarly, [WHAG15] combines this setup with a simple 3-axis CNC-mill to provide a more flexible design process that allows objects to iteratively evolve through both digital and physical input, in a process that they call *bidirectional fabrication*. In [ZP12], a handheld digital milling device is proposed that lets the user freely work on a piece of material, only controlling the spindle's speed and retraction of the shaft. In the context of woodworking, there is a new, vision-controlled device [Sha18], bridging the gap between large-scale CNC-mills and handheld routers.

2.1.1 Reliefs

The digital design and creation of reliefs has fostered a significant amount of research in the last twenty years, since the compression of a 3D model's

depth poses an interesting problem to the computer graphics community. Reliefs can be categorized into high, low or sunken reliefs, depending on the amount of compression. High reliefs are usually not compressed more than half of their original depth and often have some undercut areas. Low and sunken reliefs instead are usually just common height fields and only have a small offset from the base surface. In this context, we review most related work on the depth compression of 3D models and the design and creation of different types of reliefs.

Height field compression. The digital generation of bas-reliefs was pioneered by Cignoni et al. [CMS97], who created bas-relief models of given 3D objects by linearly compressing (squeezing) the depth map of their rendering, obtained using OpenGL-based rasterization. By swapping the linear compression with more advanced nonlinear and adaptive scaling functions, it is possible to increase the visual quality [SRML09]. Nonetheless, there is no direct connection between the compression of the geometry and the lighting equation: the surface normals may significantly change after the squeezing operation, and the resulting bas-reliefs are prone to looking different from the desired appearance, requiring a heuristic post-processing step to add details and increase the depth illusion.

Gradient field compression and Poisson reconstruction. A breakthrough in the generation of digital bas-reliefs has been proposed in [WDB+07], where instead of compressing the height field directly, modifications are applied to its surface gradients, and a new height field matching the manipulated gradients in the least-squares sense is extracted by solving the Poisson equation (a linear system). Many variants of this algorithm have been explored [SBS07, KTB+09, BH11, WCKZ12, ZZZY13], with different kinds of filters applied to the gradients, or to the final surface in post-processing. However, all these methods suffer from the intrinsic limitation that the modified gradients are in general not integrable, and the normals of the surface generated in the Poisson step can be far from the desired normals.

Laplacian compression. Ji et al. [JMS14] propose to directly minimize the L_2 difference between the Laplacian of the bas-relief height field and the depth discontinuity free surface computed from a rendered normal image. This approach allows for artistic editing in the 2D domain and produces higher-quality results than previous methods. However, their formulation is limited to height fields and does not provide exact pointwise control over the depth of the generated bas-reliefs. They use a penalty-based approach to

control the thickness of the height field which depends on multiple parameters. Furthermore, they only consider an orthographic projection in their optimization. Our approach in Chapter 3 lifts both limitations, providing realistic results under perspective viewing, as well as precise and fine-grained depth/volume control. Concurrent work to ours [ASH15] only uses a sparse set of user-defined attenuation points within the scene to formulate a similar deformation problem. It therefore lacks full control over the final desired depth range which needs to be fixed by a post-process range compression step.

Encoding the height field and depth control. The majority of the methods mentioned above encode the height field as an image, and only few methods work directly on the input 3D geometry representation. While the former approach greatly simplifies the implementation of the algorithm, the latter approach allows to preserve the sharp details in the scene and the original modeling resolution. None of the existing methods allow fine-grained control of the depth: they provide a parameter that controls the maximum depth of the bas-relief, but they cannot be used to create bas-reliefs that fill a complex volume shape. Our method (Chapter 3) can guarantee that the sculpture will be contained within a specific depth volume, specified as a per-vertex range, and it optimally uses the entire available space.

Bas-relief ambiguity. If a surface with Lambertian reflectance is viewed orthographically from a fixed view point, there is a set of transformations of the object's geometry and the corresponding light sources which do not change the perceived shading and self-shadowing of the object, making it impossible for an observer to determine its true geometry. This is known as the "Generalized Bas-relief Ambiguity" (GBA) [BKY99] and even holds for slight changes of the viewpoint. Unfortunately, this is not applicable in real world scenarios where the perspective has to be taken into account and in general no assumptions can be made about the position and direction of the illumination. Chandraker et al. [CKK05] and Tan et al. [TQZ11] analyze (inter-)reflections and show how they can be utilized to overcome the GBA to recover the geometry from photometric stereo where the light source directions and strengths are unknown.

Inverse relief problem. Several works considered the inverse problem related to relief creation. Zatzarinni et al. [ZTS09] extract the relief layer from scanned artifacts using robust height function fitting for archeological analysis purposes. Kolomenkin et al. [KLST11] reconstruct a fitting bas-relief

surface of certain thickness given completely flat input in form of line drawings. They employ Laplacian-based surface inflation, where the Laplacian vectors are hallucinated from the given curves.

2.2 Forming and bending

Bending is a physical process where a piece of ductile material, usually metal, is permanently deformed by an external force to match a target shape. Besides many other applications in the industry, it is extensively used to manufacture lightweight but robust boxes, shaped metal pipes and parts or even coins. But it also has many interesting applications in arts and crafts, especially if combined with algorithmic tools, which we will review in the following.

2.2.1 Wire bending

In [IIM12] a interactive computational system for the design and construction of 3D beadwork is proposed, which computes a step-by-step instruction to bend a series of connected wires, holding the beads, in order to approximate the target model. A stable self-supporting wire sculpture can be constructed using a 2D wire bending machine [MLB16]. The method allows the user to explore possible designs to approximate the target shape without the need of additional connectors for crossing wires. In [GSD+14] a new optimization approach allows to approximate a given 3D shape by a wiremesh that can be used to fabricate a feasible physical model with the help of a support structure.

2.2.2 Sheet bending

Usually, a flat sheet of material made from plastic, leather or metal can easily be bent, but not stretched without extensive forces. This significantly limits the space of feasible shapes. The authors of [KCD+16] present a method which extends this space to doubly-curved surfaces by introducing a specific pattern of cuts in the sheet, which makes it auxetic. In the follow up [KPCP18], this concept is used to design deployable structures which approximate a doubly-curved target surfaces via inflation or gravitational loading. Similarly, [WPGS18] cuts packable springs from flat material sheets, such that a given optimized 3D model can be approximated by simply stretching these springs onto a support structure.

2.2.3 Thermoforming

Thermoforming is an industrial process used to fabricate a large part of the objects we use daily, such as food packaging, disposable plates, blister packaging, plastic toys and interior paneling. The thermoforming process deforms a plastic sheet, forcing it to assume the shape of the desired mold. First, the sheet is heated until it transitions to a pliable state, and then air pressure or a vacuum is applied, such that the sheet tightly adheres to the mold. Several commercial software solutions [Acc16, Rhe16, ESI16] can simulate the thermoforming process by using advanced non-linear FEM models for plastic sheets [NTD90, KBV92, KGVM97]. Differently from our approach in Chapter 4, they are particularly interested in the thickness of the sheet after deformation, which determines the robustness and material properties of the fabricated good. They rely on a large set of parameters that are difficult to find without physical material tests [MDBI15]; the tests are expensive and must be performed on each set of machines and materials used, making them feasible only for industrial production.

Crowdfunding projects from the maker community are popularizing affordable devices like FormBox [May18] or Vaquform [Inc18], which enable private users to use them at home. But the creation of a heat resistant mold from a digital model remains a challenge, since most low-price 3D printers use a plastic material that cannot sustain the heat of the thermoforming process. Therefore, works from J. Yamaoka [YK16, YK17] propose an actuator controlled forming surface that allows to quickly create new 2.5D mold shapes for vacuum forming.

Interestingly, the thermoforming technique can also produce highly detailed color-textured objects if an image is printed onto the plastic sheet prior to deformation (see Section 2.10).

2.3 Casting and molding

Casting or molding is a process in which a liquid material is poured in a mold, that encloses an empty space with the shape of the target object. Once the liquid material is filled in and has hardened, the mold can be taken apart to release the resulting cast. There exist a variety of different casting and molding techniques, involving various materials. It is usually used to make complex shapes that would be difficult to make otherwise. Casting has been used for thousands of years [Rav06] to create sculptures, jewelry, weapons and many tools. It is used in the industry, e.g. for mass production, but also

from makers and designers for rapid prototyping or fabrication of single pieces. The challenge usually lies in the design of a feasible mold, especially if used multiple times to fabricate complex shaped parts. If no special care is taken, the mold can get stuck easily to the casted object in regions with undercuts or fine geometric features.

In [HAM14] and [HMA15], the authors describe methods that slightly deform a target shape to enable the automatic creation of a rigid mold from multiple pieces, such that they can be separated after casting without any problems. Another approach from L. Malomo et al. [MPBC16] creates a flexible, deformable shell mold, using a 3D printer, which can be simply opened along an optimized cut to reveal the final object. In [NAI+18], an interactive optimization technique to create thin-shell casts from 3D models is proposed. After decomposing the shell into moldable segments, pairs of rigid mold pieces are computed that can also be used for professional injection molding machines. T. Alderighi et al. [AMG+18] create so-called meta-molds to cast reusable silicone molds for complex shapes. Metamolds are designed through a novel segmentation technique and only requires of-the-shelf materials. Besides rigid objects, molding can also be used to cast deformable materials like rubber or silicon. In the work of M. Skouras [STBG12] a fabrication-oriented design and physics-driven shape optimization method for custom-shaped rubber balloons is proposed which, after casting, can be inflated to approximate the target shape.

2.4 Paper crafts

Papercraft includes a collection of art forms that use paper to form objects of various shapes. Techniques like cutting, tearing, gluing or layering help to work and form the material. Since paper is relatively inexpensive, comes in many colors, is easy to work with and readily available, it is one of the most popular materials for hobbyists and children all over the world and is also often used in artistic work.

2.4.1 Paper models

Creating three-dimensional objects from paper is far from trivial. Different approaches show how to compute a decomposition of a target surface into developable patches which can then be cut and assembled by folding and gluing [MS04, STL06, MGE07, SP11, TWS+11, TBWP16, PDRhK18, SGC18].

Alternatively, the shape can be a priori modeled or approximated as a piecewise developable surface, which is a current topic of active research [KG02, RSW+07, JHR+15, FBR+17, KFC+08, LLH09, ZLCY12, CKK+15, TBWP16]. Since paper sheets have many similarities to thin pieces of fabric, the former approaches are relevant there as well (see Section 2.5).

2.4.2 Origami folding

Even more difficult is the art of creating 3D shapes from paper if neither cutting nor gluing is allowed, but only folding. This technique is called Origami and only few related works are capable of approximating a given 3D target shape [Tac10, KFC+08].

2.4.3 Pop-ups construction

Pop-ups are flat paper constructions which can be transformed into a surprising 3D shape by a single move. The most popular examples are pop-up cards for various occasions like birthdays or weddings. Designing them is challenging and requires skills and experience. [JLYL14, LJGH11] propose methods to compute and fabricate such pop-up cards for a given target model. If this construction is limited to be cut out of a single sheet of paper, it is called *origamic architectures* and has been studied in [LSH+10, LLL+14, MS04].

2.4.4 Paper weaving

Weaving is usually a fabrication method for the production of fabrics from yarns that are interlaced at right angles. But a similar technique can be used with paper stripes to construct free-form surfaces to approximate target shapes [TWZ+17, TISM16a].

2.5 Fabric crafts

In comparison to paper, fabric is usually more flexible and can be stretched and sheared depending on the type of material. It is either made from natural or artificial fibres through different techniques, like weaving, knitting and others. Objects made of fabrics are ubiquitous in our everyday life: cloths, shoes, curtains, bags, furniture and many more. Due to its flexibility, fabric can be challenging to work with. A dress looks different in motion, worn by a person, than on the drawing table. Closed objects like pillows need to be

stuffed to keep their round shape without support. Therefore, designing with fabric is often an incremental process and pieces need to be refined multiple times until the final result looks right. Like in Section 2.4 about paper models, to fabricate a certain target shape, methods were derived to approximate them by pieces of developable surfaces, typically achieved by segmenting or cutting the shape into parts with low Gaussian curvature and parameterizing each part onto the plane ([MI07, II08, JKS05, Wan10, MAWS15]). Foils can often be used interchangeably with stiff fabric and allow the designs of inflatable structures [STK+14]. The assembly by gluing or sewing requires precision and carefully following the instructions. In contrast, in Chapter 5 we propose a method using a single zipper tape that makes the final assembly trivial and a nearly mindless task (see also Section 2.9).

2.5.1 Zipper-curve design

In the context of designing zipper-curves, we give a brief overview on related methods and compare their properties to the proposed approach in Chapter 5 in Table 2.1. Finding a 2D shape that *perfectly* reproduces a 3D mesh can be achieved using so-called *mesh stripification* algorithms, which cut the mesh into triangle strips [Ros99, O'R15, LDD+10, EG04]. However, in addition to being highly mesh dependent, the resulting strips have many sharp turns that make attaching a zipper and manipulating the zipper slider difficult, if not impossible, and result in an uneven strip width. A recently granted patent proposes a method for approximating shapes by developable surfaces passing through *labyrinths* on the surface [Ped11] (see also a related paper [PS06]), reminiscent of space filling curves. While these curves are smooth and uniform, they bend excessively, which would make attaching a zipper very hard if not impossible.

	Stripification	Paper craft	Labyrinth	Ours
Single part	✓	x	✓	✓
Uniformity	x	✓	✓	✓
Low curvature	x	✓	x	✓

Table 2.1: *Comparison of alternative curve design approaches.*

2.6 Knitting and Stitching

Before modern times, most clothes were hand-made by knitting using yarns from wool, cotton or silk. Nowadays, knitting is still one of the most used

techniques in the industry, besides weaving, to produce various fabrics using highly automated machines. To design and create suitable instructions for complex 3D target surfaces, J. McCann et al. [MAN+16] proposed a tool to compute machine instructions for assemblies of high-level shape primitives. The methods of [NAH+18, WGF+18] go one step further and can create knitting instructions for arbitrary 3D surface models. S. Hudson [Hud14] developed a novel stitching device that can fabricate volumetric objects and shapes from yarn, similarly to a 3D printer.

2.7 From planar pieces

Approximating 3D shapes with planar pieces is a fast and inexpensive option to fabricate objects from different sheet material. It is the bread and butter of many architectural students who spend hours assembling their models from layers of laser-cut sheets of wood. This technique is related to so-called *laminated object manufacturing* (LOM) [MBW16]. Like 3D printers, there are special devices for paper [Ltd16] or fabric like materials [PMHM15] available. In [HBA13] a method is proposed to decompose a shape into segments with different slicing orientations and show that this is superior to slicing the whole shape in only one direction. A variety of other methods approximate a target shape by interlocking planar pieces [MSM11, HBA12, SP13, MUS14, ZGPR16, CPMS14, SCGT15]. They can be used, e.g. in modelmaking and rapid prototyping, to create a scaffold structure as the basis for the final fabricated model. Others use different types of connectors to attach planar parts to assemble the hull of a arbitrary 3D object [CSLM13, RA15, AKW+16].

2.8 Special building blocks

Numerous approaches use special building blocks to approximate 3D shapes. They are either based on parts which are commercially available or derive a method to overcome fabrication related limitations or to create 3D puzzles.

2.8.1 Manufacturable parts

[SFLF15] addresses the fabrication devices' building volume limitations and propose an interlocking block decomposition to fabricate larger models. Other methods [HLZC14, HAM14, MLS+ar] compute specific height-field decompositions to create parts which can be better fabricated with classical manufacturing techniques such as mold casting or milling.

2.8.2 Puzzles

Variations of interlocking piece puzzles that approximate given 3D models can be designed with the following methods: [LFL09, SFC12, XLF+11, Séq12].

2.8.3 LEGO blocks

LEGO blocks can be used to approximate arbitrary 3D shapes [TSP13, KTM16], which should be stable under their own weight [WW12, HWS+16]. In [LYH+15] the authors use centroid adjustment and inner engraving to create balanced LEGO sculptures. Similarly, a method to automatically create brick sculptures from pixel arts is proposed in [KLC+15].

2.8.4 Zometool

The Zometool is a commercially available, versatile construction set made of small plastic connector nodes and struts of various colors. It has been used for architectural designs and in various research fields to visual mathematical objects or structures like molecules. H. Zimmer et al. [ZLAK14, ZK14] propose a method to approximate free-form surfaces with Zometool structures.

2.8.5 Tensegrity

Tensegrity is a 3D construction approach with *tensional integrity* - rigid elements like bars or struts are in compression inside a net of tensioned links (e.g. cables) without touching each other. These structures can be of exceptional rigidity and have been used successfully to build bridges or playground climbing structures for kids. D. Gauge et al. [GCMT14] developed a computational design tool for creating physical characters from simpler tensegrities, used as building blocks. In [PTV+17], N. Pietroni et al. propose a method to approximate an arbitrary input shape with a stable tensegrity structure.

2.9 Simplifying assembly

Previously discussed works rarely consider the actual challenges encountered in the final physical assembly process of a computed design. E.g. in papercraft, freehand gluing or taping together multiple pieces is a very tedious and time consuming process. Recently, there has been some interesting works that tackle this problem of simplifying and speeding up the assembly process.

Similar to our method in Chapter 5, [SP06] propose a papercraft construction approach that computes a single, possibly self-overlapping unfolding, that supersedes the usual part identification instructions. But attaching a zipper would be difficult, since the cuts can be very jagged. Another papercraft method [TWS+11], uses colors to help with the identification of matching borders in the construction process. Several works suggest ways to automate the assembly of paper models from flat cut-outs by using strings [KMM17] or temperature triggered self-folding materials [AMT+14, ATG+18, TFM+14] with collision-free linear folding paths [HKL18]. R. Guseinov et al. [GMB17] uses two pre-stretched membranes to force a printed structure to take shape in 3D once released. K. Wolff et al. [WPGS18] derived a method to cut thin, planar spirals out of flat panels which can be simply pulled apart to take on the shape of a 3D spring whose contours approximate a target model.

2.10 Surface color texturing

Manufacturing color-textured objects is an important long-standing problem that has been tackled with many technologies in the last decades. We provide a brief overview about the state-of-the-art methods proposed both in academia and in the industry.

Color 3D printing. Powder 3D printers can produce textured objects by mixing colored binders during the printing process [3ds16]. Similarly, paper printers can produce objects by glueing printed sheets of paper together [Ltd16]. Both technologies require expensive dedicated 3D printers and long printing time [RBK+13, VWRKM13, HL14, RCM+14, Cut15].

Hydrographics. Texture can be transferred onto an object after its fabrication using water transfer printing, or hydrographics [ZYZZ15, PDP+15]. The texture image is printed on a polymer sheet that dissolves in water, leaving the ink floating on its surface. The object is then dipped into the water to transfer the ink onto it. A limitation of this technique is that flat regions parallel to the water surface cannot be properly colored due to the formation of air bubbles. Our method in Chapter 4 shares some similarities with hydrographics, but thanks to the use of thermal transfer paper instead of water, the colors are more vivid and flat parts are not problematic. However, our method is limited to plastic materials and cannot be used to fabricate colored wood or glass objects.

Projection. Time-varying texture can be applied to objects that undergo rigid transformations or non-rigid animation known in advance using projectors [LWN+09, RWLB01, BBG+13]. However, this technique only temporarily produces the colored appearance.

Piecewise mapping In [WK17] a piecewise covering with 2D images by gluing is proposed to color a 3D object. But the process is highly manual and only very simple, smooth surfaces without many geometric details are feasible. For these cases, segmentation methods from Section 2.4 and Section 2.5 could also be employed.

Assisted painting. A few recent techniques propose to paint a flat canvas using a robot [LPD13] or computer controlled spraying [SMPZ15, PJJSH15]. These techniques are currently restricted to flat surfaces, and it would be challenging to extend them to paint on a complex shape.

Industry solutions. An common industrial approach to textured thermoforming [The16] requires thermoforming the desired shape with a special calibration texture and then 3D scanning it. Correspondences between the flat texture and the formed 3D surface are then extracted, and the resulting mapping is employed to apply arbitrary images and text onto each subsequent copy of the same 3D object. This approach is limited to simple geometries that are easy to scan.

Thermoforming. Concurrent to our work [tPG+16], the method of Y.Zhang et al. [ZTZ17] proposes a similar solution for coloring thermoformed objects. In contrast to our approach they assume a fixed set of model parameters and have not calibration step to account for differences in the thermoforming setup.

2.11 Parameterization.

In the context of the method proposed in Chapter 5, we also need to give a short review about mesh parameterization, which is an extensively studied topic, see e.g. the survey in [HLS07]. The more recent relevant parameterization literature is presented in [SS15, KGL16, RPPSH17]. In Chapter 5 we introduce a *new type* of global parameterization, which extends cylindrical parameterization [Tar12]. Global parameterization is primarily used for quad

meshing, where parts of seams in the parameter domain are related to each other by a rotation of integer multiples of $\pi/2$. A recent review can be found in [BLP+13]. In our specific case, we require a different form of seamless mapping, based on cylindrical domains [TBTB12, RLL+06, MCK08, KCPS15]. Our main inspiration is [KNP11], where the authors propose an approach for drawing stripes on *tubular* shapes that can be used for generating textures (see also [LAPS17]). Their approach is based on a seamless parameterization aligned with a 2-RoSy field obtained from the principal curvature directions. This causes problems near umbilical points, where the principal directions are unstable. In contrast, we employ a parameterization based on distortion minimization, which avoids this problem, and generally leads to less distorted mappings [MZ12].

CHAPTER 3

Relief creation from digital 3D models

3.1 Overview

Reliefs are surfaces whose normals resemble the normals of a different surface or a general 3D scene, giving a (false) impression of depth when observed from the right viewpoint. Reliefs have been used for centuries in artistic masterpieces, and are ubiquitous on coins and medals. The most common type of reliefs are bas-reliefs, thin layers of stone or ceramic covering a single object, but they can also be fragmented into disconnected slices to obfuscate the shape, creating interesting optical illusions (Figure 3.2).

The design of bas-reliefs has been a subject of interest in computer graphics in the past two decades. A bas-relief is essentially a 2.5D image, which has a strong relation with the depth buffer used in the standard graphics pipeline. Most works have proposed to create bas-reliefs from given digital 3D scenes by either directly compressing the depth buffer of the scene's rendering or by working in the gradient domain, where the final model is obtained by solving a Poisson equation.

In this work, we define *appearance-mimicking surfaces* (AMS) that generalize bas-reliefs, lifting their restriction to a height field. Our generalization makes the reliefs usable at a wider range of viewing angles, while still guaranteeing self-intersection free results, which is mandatory for subsequent fabrication.

Specifically, we develop a mathematical framework to compute surfaces whose normals optimally approximate the normals of a given 3D shape or

Figure 3.1: *A collection of appearance-mimicking surfaces generated with our algorithm.*

scene, while strictly obeying given depth- or volume-confinement constraints. Direct fitting of normals and spatial constraints is in general a challenging, nonlinear problem, which led previous works to employ heuristics that circumvent difficult numerical optimizations. Unfortunately, giving up the constrained optimization of normals means forfeiting bounds on geometry and appearance distortion in the resulting relief. Instead, we propose a novel view-dependent surface representation which allows us to cast the optimization as a quadratic program. The resulting problem formulation is convex, and we are guaranteed to find the optimal solution under feasible constraints.

Differently from previous works, our method does not rely on rasterization of the input geometry and the depth buffer. AMS are generated by deforming the input mesh without modifying its connectivity, thereby increasing the algorithm's efficiency, details preservation and allowing to easily transfer surface attributes. As a positive side effect of our representation, we can exactly satisfy per-vertex depth constraints and we can "project" the target shapes on disconnected and arbitrarily shaped surfaces, as shown in Figure 3.2.

Our algorithm is controllable and robust, enabling to design complicated appearance-mimicking surfaces with minimal user effort. We test our method in a variety of applications, such as the design of optical illusions in architectural settings and the creation of carving patterns on complex geometries. To verify the realism of our model and lighting assumptions, we validate our results via 3D printing.

Figure 3.2: *Inspired by street artwork painted over the steps of a staircase, we use our algorithm to embed a 3D model of an owl into a staircase. Constrained to a thin layer, the relief does not affect the function of the staircase, while being much more resistant than paint to erosion and aging.*

3.2 Method

An appearance-mimicking surface is a surface that *looks* similar to another from a fixed perspective, while having a different geometry.

Lighting model. Assuming a Lambertian material with directional lights and no specular component, we can model the color of a surface point $\mathbf{p}$ as:

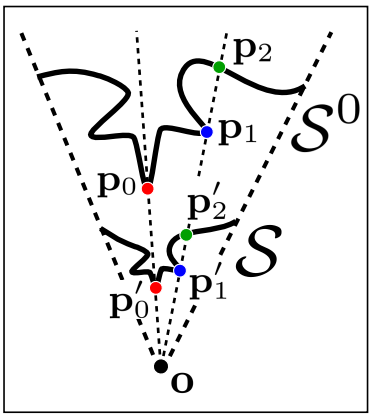
$$I_{\mathbf{p}} = k_a i_a + \sum_{\mathbf{l} \in \text{lights}} (\mathbf{l} \cdot \mathbf{n}_{\mathbf{p}})\, i_d \tag{3.1}$$

where k_a is an ambient reflection coefficient, i_a is the ambient color of the material, $\mathbf{l}$ is the light direction, $\mathbf{n}_{\mathbf{p}}$ is the surface normal at point $\mathbf{p}$ and i_d is the diffuse color. In this setting two meshes will result in identical renderings if for each point (or pixel) on the view plane the angle between the

Figure 3.3: *Appearance-mimicking surface of the* Dragon Head *model constrained to carve a V-shaped, thin geometry.*

corresponding normal on the surface and the given light direction is identical. Belhumeur et al. [BKY99] defined this as the "Bas-Relief Ambiguity" for the orthographic case and formulated a set of invariant transformations of the surface geometry and the corresponding light sources. In real world scenarios the lighting direction $\mathbf{l}$ is often not known in advance and difficult to control (e.g. the sunlight). Therefore, if an object is confined to a smaller space, we are looking for a deformation of the geometry which tries to preserve the surface normals to minimize the visual difference under various illumination conditions.

View-dependent surface similarity. We chose to constrain each point $\mathbf{p}'$ of the deformed surface S to stay on the ray emanating from a viewpoint $\mathbf{o}$ in the direction of $\mathbf{p}$ (see inset). This representation naturally preserves the surface normals under uniform scaling of the geometry for a fixed perspective. It allows us to define a surface similarity $d(S, S^0, \mathbf{o})$ to measure the perceived difference of the surface S^0 and its deformed state S when observed from a fixed viewpoint $\mathbf{o}$:



$$d(S, S^0, \mathbf{o}) = \int_S \left\| \mathbf{n}^S_{\phi(\mathbf{p},\mathbf{o})} - \mathbf{n}^{S^0}_{\mathbf{p}} \right\|^2 d\mathbf{p}. \tag{3.2}$$

Here, $\phi(\mathbf{p}, \mathbf{o})$ denotes the pointwise identification of the surface S^0 with its deformed version S. Note that we integrate over S to incorporate the change in the deformed surface area. In this work, we use a variational approach to compute an appearance-mimicking surface that minimizes the distance d, given user-provided thickness constraints.

Surface discretization. We represent the surface S as a triangle mesh $\mathcal{M} = \{\mathbf{V}, \mathcal{F}\}$, where $\mathbf{V}$ is an n-by-3 matrix that stores the coordinates of the vertices and $\mathcal{F}$ is an m-by-3 matrix encoding the connectivity. We can equivalently represent the ith vertex $\mathbf{v}_i$ of S as:

$$\mathbf{v}_i \;=\; \mathbf{o} \;+\; \|\mathbf{v}_i - \mathbf{o}\| \frac{\mathbf{v}_i - \mathbf{o}}{\|\mathbf{v}_i - \mathbf{o}\|} \;=\; \mathbf{o} \;+\; \lambda_i \frac{\mathbf{v}_i - \mathbf{o}}{\|\mathbf{v}_i - \mathbf{o}\|}, \tag{3.3}$$

where $\lambda_i = \|\mathbf{v}_i - \mathbf{o}\|$. Without loss of generality, we can assume that $\mathbf{o} = (0, 0, 0)$ and simplify Eq. (3.3):

$$\mathbf{v}_i = \lambda_i \hat{\mathbf{v}}_i, \quad \text{where } \hat{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{v}_i\|. \tag{3.4}$$

Fixing the directions $\hat{\mathbf{v}}_i$, the positions of the vertices of $\mathcal{M}$ can be expressed as a vector $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, ..., \lambda_n\}$. In this representation, all vertices defined by a choice of λ_i will project to $\mathbf{v}_i$ if seen from the viewpoint $\mathbf{o}$. Expressing $\mathcal{M}$ and $\mathcal{M}^0$ in the same parametrization, i.e. if both of them are represented as a set of some λ_i, Eq. (3.2) can be discretized as:

$$d(\mathcal{M}, \mathcal{M}^0, \mathbf{o}) = \sum_{i \in \mathbf{V}} A_i \|(\mathbf{n}_i - \mathbf{n}_i^0)\|^2, \tag{3.5}$$

where A_i is the Voronoi area associated with the ith vertex.

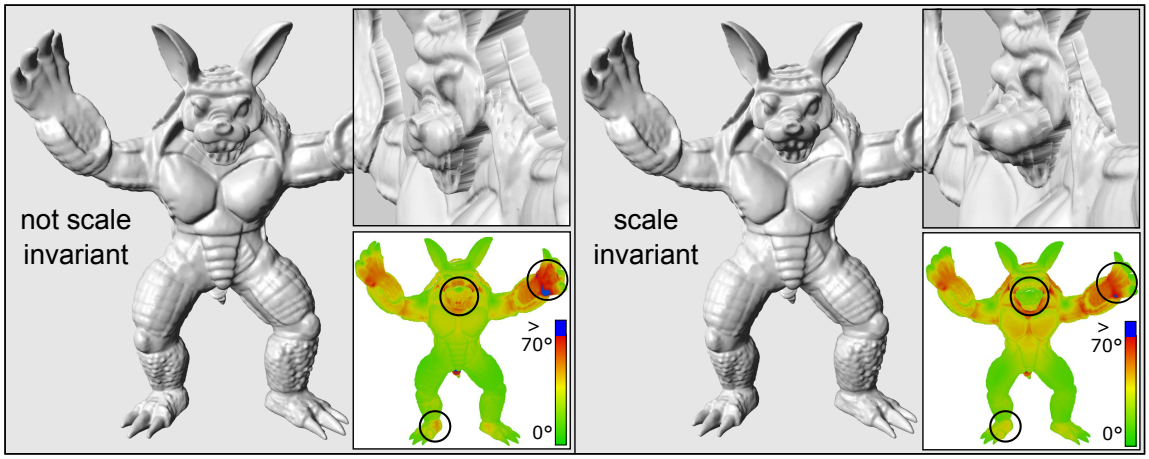


Figure 3.4: *Using a scale-dependent Laplacian (i.e., omitting the term λ_i/λ_i^0 in Eq. (3.7)) introduces artifacts (left) that disappear when using our formulation (right). The colored insets visualize the angular difference of the normals between the deformed and the initial surface.*

Linearization. We can write the surface normals as a function of the vertex positions using the discrete Laplace-Beltrami operator:

$$d(\mathcal{M}, \mathcal{M}^0, \mathbf{o}) = \sum_{i \in \mathbf{V}} A_i \left\| \frac{(\mathbf{L}\mathbf{D}_{\boldsymbol{\lambda}}\hat{\mathbf{V}})_i}{H_i} - \frac{(\mathbf{L}^0\,\mathbf{D}_{\boldsymbol{\lambda}^0}\hat{\mathbf{V}})_i}{H_i^0} \right\|^2, \tag{3.6}$$

where $\mathbf{L}, \mathbf{L}^0$ are discrete Laplace-Beltrami operators of $\mathcal{M}, \mathcal{M}^0$ and H_i, H_i^0 are the discrete mean curvatures at vertex i of $\mathcal{M}, \mathcal{M}^0$, respectively; $\hat{\mathbf{V}}$ is the n-by-3 matrix stacking all $\hat{\mathbf{v}}_i$s and $\mathbf{D}_{\boldsymbol{\lambda}}$ is a diagonal matrix with entries λ_i on the diagonal (and similarly for $\mathbf{D}_{\boldsymbol{\lambda}^0}$). The notation $(*)_i$ means that we extract the ith row of $*$. For more details about the definition of the discrete Laplace-Beltrami operator and the mean curvature we refer the reader to [BKP+10].

Similarly to previous deformation algorithms [BS08], we linearize this expression by replacing the area weighting and the Laplacian of the deformed mesh $\mathcal{M}$ with the corresponding quantities and operators of the original mesh $\mathcal{M}^0$.

$$d(\mathcal{M}, \mathcal{M}^0, \mathbf{o}) = \sum_{i \in \mathbf{V}} A_i^0 \left\| \frac{(\mathbf{L}^0\mathbf{D}_{\boldsymbol{\lambda}}\hat{\mathbf{V}})_i}{H_i^0} - \frac{(\mathbf{L}^0\,\mathbf{D}_{\boldsymbol{\lambda}^0}\hat{\mathbf{V}})_i}{H_i^0} \cdot \frac{\lambda_i}{\lambda_i^0} \right\|^2. \tag{3.7}$$

The scaling factor λ_i/λ_i^0 compensates for the change in scale of the Laplacian vector due to the linearization (see Figure 3.4). Introducing this factor makes d invariant to uniform scaling of $\mathcal{M}$, similarly to the scale-invariant Laplacian deformation energy proposed in [SCOL+04]. The main difference

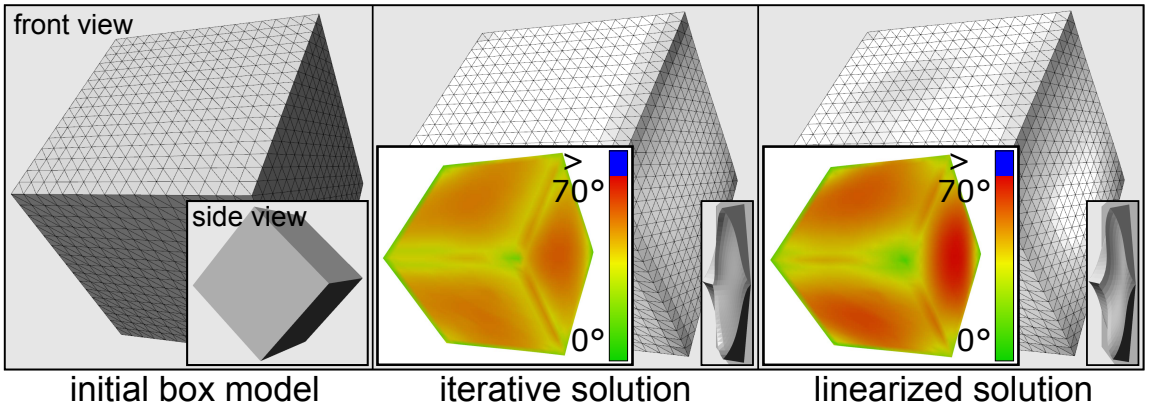


Figure 3.5: *The depth of the* Box *model (1538 vertices) is constrained to a small range. The angular difference of the normals (color insets) introduced by the linearization in Eq. (3.7) (right) is higher for low-curvature regions than in the more accurate solution computed by iteratively updating the cotangent Laplacian (middle).*

is that in our parametrization, the local scaling is given in closed form, as opposed to the local least-squares fitting of [SCOL+04]. In our case, scaling λ_i induces a uniform scaling in the neighborhood of λ_i (see Eq. (3.4)). The scale invariance introduces rank deficiency in the optimization, but can be fixed by constraining the λ_i of a single vertex i. This can be modeled as an equality constraint:

$$\mathbf{C}_E \boldsymbol{\lambda} = \mathbf{b}. \tag{3.8}$$

Linear approximation effect. The linearization error introduced in Eq. (3.7) is higher in areas with low curvature, as shown in Figure 3.5, where the depth of the *Box* model is constrained to a small range. To compare, we iteratively computed the more accurate solution of the nonlinear problem, updating the cotangent Laplacian of the deformed surface in every iteration. Unfortunately, this approach works only for very small and regular meshes and does not converge for any other presented result. The reason is that the cotangent approximation of the Laplacian becomes increasingly inexact for triangles with angles exceeding 90°. Therefore, we opt for using the less accurate but much more stable and efficient linearization in Eq. (3.7).

Bias for high frequency details. Eq. (3.7) is challenging to minimize numerically, due to the extreme variation in the range of the term H_i^0. The curvature is close to zero in all flat or very smooth parts of the mesh, introducing an

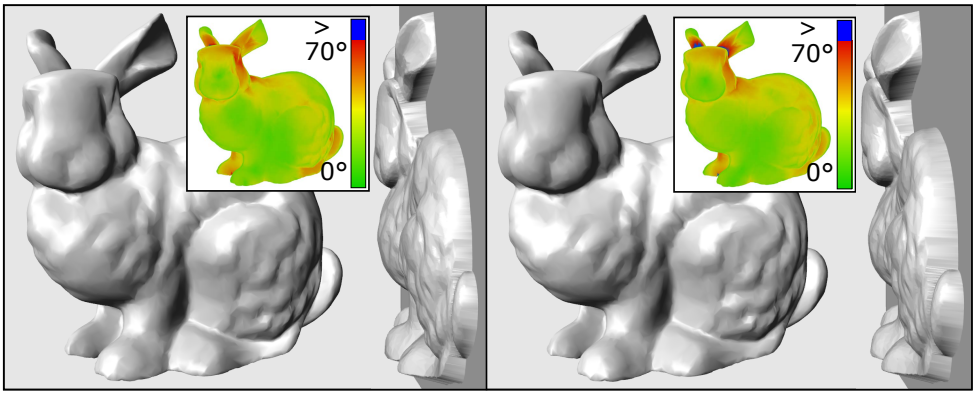


Figure 3.6: *Ignoring the mean curvature term has a minimal effect on the results and makes the optimization numerically stable. The insets show the angular difference of the normals between the deformed and the initial surface.*

extreme scaling, which leads to numerical problems. We remove this instability by adding a weighting that biases the error measure towards preserving high-curvature details. We weigh the norm of vertex i with $(H_i^0)^2$, hence giving less importance to the unstable flat regions and canceling out H_i^0:

$$d(\mathcal{M}, \mathcal{M}^0, \mathbf{o}) = \sum_{i \in \mathbf{V}} A_i^0 \left\| (\mathbf{L}^0 \mathbf{D}_{\boldsymbol{\lambda}} \hat{\mathbf{V}})_i - (\mathbf{L}^0 \mathbf{D}_{\boldsymbol{\lambda}^0} \hat{\mathbf{V}})_i \frac{\lambda_i}{\lambda_i^0} \right\|^2 \tag{3.9}$$

The effect of introducing the bias is minor, as shown in Figure 3.6, but makes the optimization numerically stable.

Depth constraints. Eq. (3.9) is quadratic in $\boldsymbol{\lambda}$ and can be efficiently minimized by solving a linear system. The thickness of the resulting surface is completely controlled by $\boldsymbol{\lambda}$, which can be easily bounded on each vertex using inequality constraints of the form:

$$\lambda_i^{\min} \leq \lambda_i \leq \lambda_i^{\max}. \tag{3.10}$$

This transforms the minimization of Eq. (3.9) into a quadratic problem, which can still be optimally solved.

Disconnected pieces. Depending on the application, it might be useful to define depth range constraints that are discontinuous. This would enable us to optimize for appearance-mimicking surfaces that are themselves discontinuous, like the pillar surfaces in Figure 3.14. This could be achieved

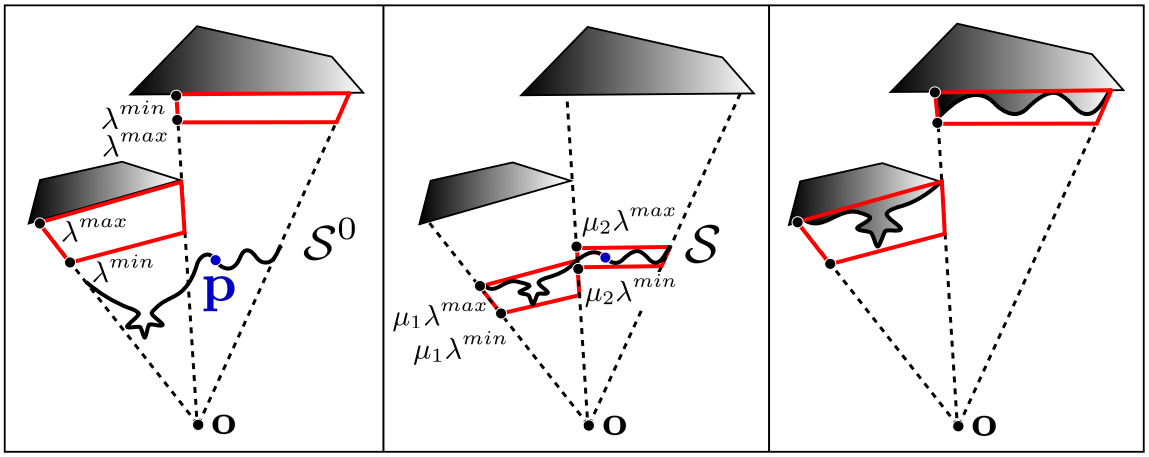


Figure 3.7: *Depth constraints can be specified independently for every disconnected component of the target surface (left). The bounds are moved during the optimization to enlarge the solution space and increase the AMS quality (middle). The optimized surface is then projected to every component, guaranteeing to exactly satisfy the original depth bounds.*

by splitting the mesh into multiple disconnected sets of vertices and solving independent optimizations. However, this approach requires remeshing and splitting the shared boundary between every group of vertices, potentially generating low-quality triangles that make any further optimization unstable. Moreover, additional constraints to match the normals for shared vertices are then needed.

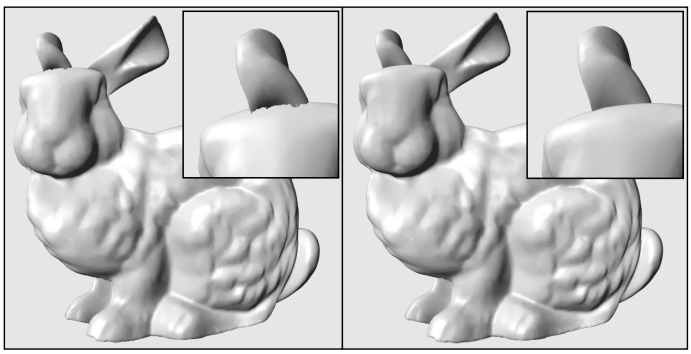
We therefore propose a different approach to directly use our algorithm, without the need to split the mesh. Instead of providing absolute lower and upper bounds $\lambda_i^{\min}$ and $\lambda_i^{\max}$ for each vertex, we define a dynamic range for each group of independent vertices, which can be freely moved during the optimization. This provides maximal freedom to optimize the energy, and only after the optimization the surface is cut into pieces and displaced according to the original, discontinuous geometry. We model this idea by adding a variable μ_g for each group g of independent vertices, transforming the constraints into:

$$\mu_g \lambda_i^{\min} \leq \lambda_i \leq \mu_g \lambda_i^{\max}. \tag{3.11}$$

We sketch an example in Figure 3.7, where the depth bounds are independently provided for each disconnected pillar (left). Our optimization finds the optimal appearance-mimicking surface (AMS) that satisfies the depth bounds up to a scaling, which is controlled by the μ s. After the optimization, the λs are scaled back by multiplying by the inverse of the μ s to warp each

piece of the AMS back to its associated pillar (right). The point $\mathbf{p}$ (in blue) is fixed to make the solution unique, as discussed previously. The result, after scaling back, is independent of the choice of $\mathbf{p}$. Note that the same formulation with only one λ is used for the case of one simple continuous depth range constraint.

Self-intersection avoidance. The deformed surface might contain self-intersections, as visible in the boundary between the head and the ear of the *Bunny* relief (see inset).



Since each vertex is constrained to move on a ray, this can only happen when two parts of the surface change their depth ordering, with respect to the viewpoint $\mathbf{o}$. To prevent self-intersections, we force vertices to preserve their depth relationships: for every vertex $\mathbf{v}_i^0$ we cast a ray from $\mathbf{o}$ in direction $\hat{\mathbf{v}}_i^0$. For every pair of consecutive hits, we add a linear inequality constraint that enforces depth ordering preservation. Note that the rays will generally hit the interior of a triangle, and the hit position can then be represented using barycentric coordinates. All inequalities can be stacked in matrix form as:

$$\mathbf{C}_I \boldsymbol{\lambda} \leq \mathbf{d}. \tag{3.12}$$

While this procedure does not guarantee the elimination of edge-to-edge intersections, we found that this is not a problem in our experiments, and it does not affect the 3D-printed results since the resolution of the printer is typically lower than the mesh resolution.

Height fields. Our formulation can be specialized to create appearance-mimicking surfaces that are height fields w.r.t. the viewpoint $\mathbf{o}$, thereby increasing the optimization efficiency and quality of the results, especially for very thin bas-reliefs and carvings.

Assume we wish to create a height field AMS from a general surface S. Many parts will not be visible from $\mathbf{o}$ due to self occlusion: Constraining such

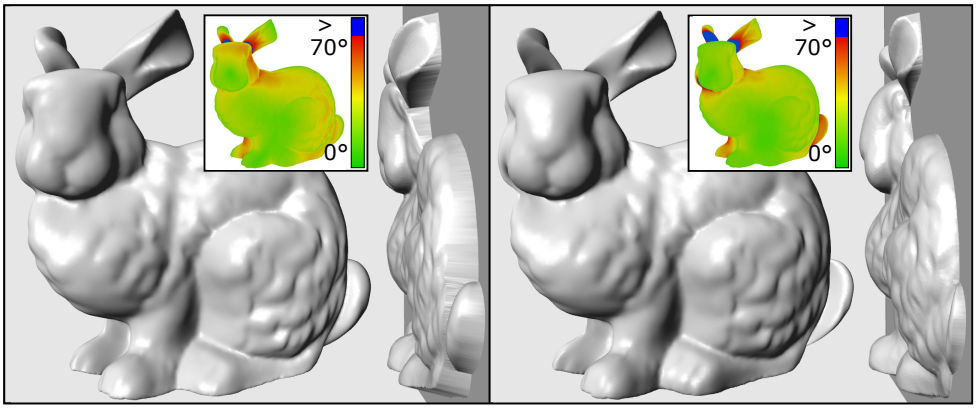


Figure 3.8: *By dampening the influence of the hidden vertices with the weights w_i, we leave more freedom to the optimization, which better preserves the surface details in the visible regions.*

vertices would unnecessarily restrict the degrees of freedom in the optimization, since the occluded vertices will not be visible. Therefore, we only set hard constraints on the visible vertices, which considerably speeds up the optimization (for the *Bunny* model used in Figure 3.9 the computation time is reduced by a factor of 3.6). If large regions of a surface are occluded, the quality can be further increased by dampening the influence of the corresponding vertices in the energy, giving more freedom to the visible parts. To model this optional feature, we introduce an additional weighting w_i that scales the difference in the vertex normals:

$$d(\mathcal{M}, \mathcal{M}^0, \mathbf{o}) = \sum_{i \in \mathbf{V}} w_i^2 A_i^0 \left\| (\mathbf{L}^0 \mathbf{D}_{\boldsymbol{\lambda}} \hat{\mathbf{V}})_i - (\mathbf{L}^0 \mathbf{D}_{\boldsymbol{\lambda}^0} \hat{\mathbf{V}})_i \frac{\lambda_i}{\lambda_i^0} \right\|^2 . \tag{3.13}$$

In Figure 3.8, we assign a weight of 1 to the visible vertices and 0.1 to the others, obtaining a better approximation of the geometric details. We enabled this additional weighting only for the surface in Figures 3.8 and 3.11. The weighting could be exposed to the users, enabling to easily control the deformation by specifying which parts are important to preserve.

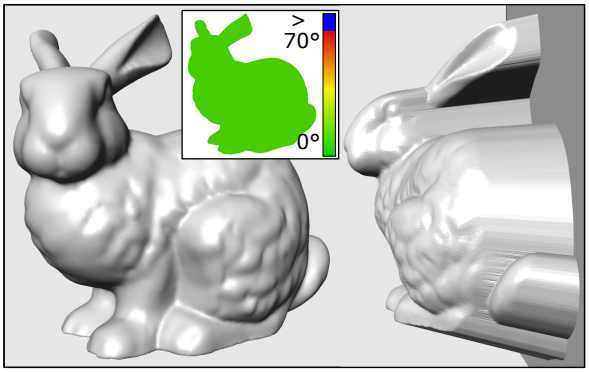
Optimization. The energy and the constraints can be written in matrix form:

$$\begin{aligned} \underset{\boldsymbol{\lambda},\boldsymbol{\mu}}{\text{minimize}} \quad & \|\mathbf{D}_A \, \mathbf{D}_w (\tilde{\mathbf{L}}^0 \mathbf{D}_{\hat{\mathbf{V}}} - \mathbf{D}_{\mathbf{L}_\theta}) \, \mathbf{S} \, \boldsymbol{\lambda}\|^2 + \alpha \|\boldsymbol{\mu}\|^2 \\ \text{subject to} \quad & \mathbf{C}_I \, [\boldsymbol{\lambda} \;\; \boldsymbol{\mu}]^\mathsf{T} \leq \mathbf{d}, \\ & \mathbf{C}_E \, [\boldsymbol{\lambda} \;\; \boldsymbol{\mu}]^\mathsf{T} = \mathbf{b} \end{aligned} \tag{3.14}$$

Where $\mathbf{D}_A$ and $\mathbf{D}_w$ are $3n$-by-$3n$ diagonal matrices containing the square roots of the areas A_i^0 and the weights w_i, respectively. $\tilde{\mathbf{L}}^0$ is a $3n$-by-$3n$ matrix and equal to $\mathbf{L}^0 \otimes \mathbf{I}_3$, where $\otimes$ is the Kronecker product, $\mathbf{D}_{\hat{\mathbf{V}}}$ is a $3n$-by-$3n$ diagonal matrix of the row-wise stacked elements of $\hat{\mathbf{V}}$, $\mathbf{S}$ is a $3n$-by-n selector matrix, coupling all λs with the x, y, z coordinates in the system and can be written as $\mathbf{I}_n \otimes [1,1,1]^T$, $\mathbf{I}_n$ being an n-by-n identity matrix. $\mathbf{L}_\theta$ is a $3n$-vector defined as follows:

$$\mathbf{L}_\theta = \mathbf{D}^{-1}_{(\mathbf{S}\boldsymbol{\lambda}^0)}\, \tilde{\mathbf{L}}^0\, \mathbf{D}_{\hat{\mathbf{V}}}\, \mathbf{S}\, \boldsymbol{\lambda}^0. \tag{3.15}$$

The regularization term on μ is necessary to make the energy gradient matrix full-rank. We set $\alpha = 10^{-7}$ in all our experiments. This regularization has an intuitive meaning: it makes the minimizer unique by selecting the smallest μs which correspond to moving the surface as close as possible to the upper end of the depth range constraint. Note that this is indeed the desired behavior, since it minimizes the thickness of the AMS on each disconnected part. The influence of the regularization can be neglected and the solution of an unconstrained optimization is identical to the initial model up to numerical errors, as shown in the inset.



unconstrained optimization

We experimentally discovered that converting our QP formulation to the equivalent conic program greatly improves performance, on average by a factor of 5. We use the multi-threaded conic solver in MOSEK for all our experiments [AA00]. See Appendix for the implementation details.

3.3 Results

We used a quad-core Intel i7 processor clocked at 3.4 GHz to compute all our results. Our prototype is written in C++/MATLAB and uses the MOSEK solver for the conic optimization [AA00]. Statistics on the meshes and on the computation times are summarized in Table 3.1. To support interactive design of our examples, we used a low-resolution version of each model (left part). We used a ZCorp 650 to 3D print our results, employing a clear binder color and default printing options.

	Interactive			Final		
Model	**#V**	**#F**	**Time**	**#V**	**#F**	**Time**
Dragon	41k	83k	51s	300k	600k	1297s
Owl	20k	41k	32s	218k	437k	760s
Dragon Head	20k	39k	28s	304k	609k	825s
Armadillo	10k	20k	28s	180k	361k	2961s
Fish	10k	19k	11s	279k	559k	920s
Bunny	10k	19k	10s	40k	79k	85s
Face	10k	19k	9s	40k	80k	42s
Cow Herd	24k	51k	9s	417k	834k	330s
Pillar Forest	22k	43k	55s	221k	444k	435s

Table 3.1: *Statistic of all the used model meshes and the times needed for the conic optimization.*

Variable depth. We compare our method with a simple linear compression in a sequence of bas-reliefs ordered according to decreasing depth. As can be seen in Figure 3.9, our approach better preserves the geometric details even for extremely thin surfaces.

Comparison with [WDB$^+$07]. We compare our results with [WDB$^+$07] in Figure 3.11, using the same model and viewpoint. We reimplemented the method of [WDB$^+$07] without the optional post-processing sharpening step and used the same triangle based discretization for both methods. The two results share many similarities, with a slight edge for our method that better preserves the fine details. One important difference is that our algorithm allows to exactly control the depth of the bas-relief without resorting to a linear scaling in the post-processing. For a physically-printed model of size 24 cm $\times$ 19 cm, the bas-reliefs protrude by only 8 mm from the baseplate.

Non-height field bas-reliefs. In contrast to existing methods, our AMSs do not need to be height fields. In Figure 3.10 we show a depth-compressed *Armadillo*, where we constrain only the visual hull vertices to exactly map to the surface of the baseplate, while letting the arm and one ear unconstrained. The transition between the two regions is not a height field from the chosen viewpoint, creating an interesting effect when observing from different angles.

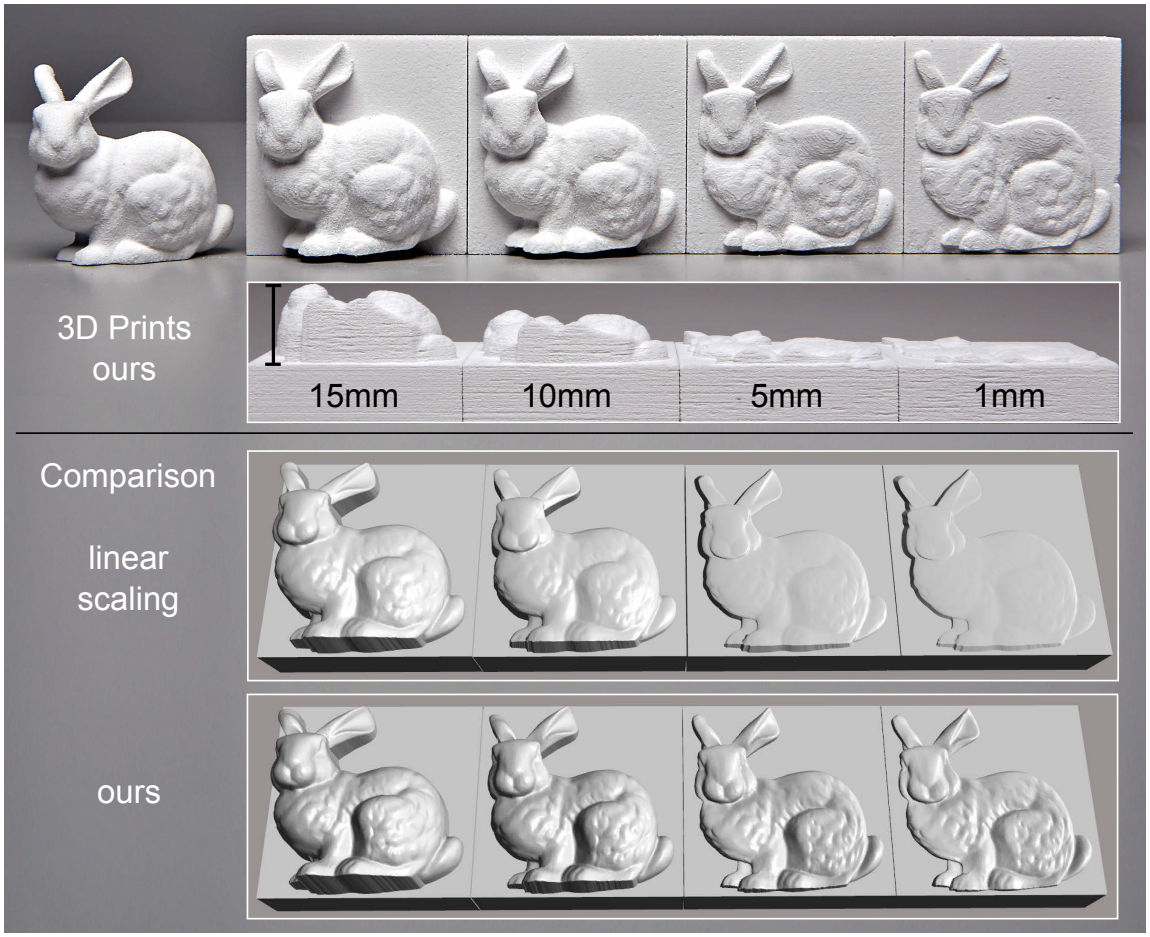


Figure 3.9: *A sequence of height restricted bas-reliefs from left to right using the* Bunny *model. Top: 3D prints generated by our method. Bottom: comparison between linear scaling and our method.*

3D Camouflage. Our method gives us exact control over the depth of each vertex. By exploiting this feature, we can constrain the AMSs to lie on many different and disconnected surfaces, as discussed in Section 3.2 and shown in Figure 3.14. We project four models from four different viewpoints onto a "forest" of pillars. While it is impossible to discern any pattern when observing from an arbitrary viewpoint, the models reveal themselves if viewed from one of the four special viewpoints. It is not trivial to manually construct such arragments of pillars, as the projected models should not interfere with the views from other viewpoints. Therefore, we implemented a simple editor which allows us to specify multiple viewpoints and pillar shapes and constantly visualizes the visible range for each of the four viewpoints in different colors as shown in Figure 3.13. The necessary depth range constraints are then automatically extracted and they guarantee intersection free views of all models within the complex "pillar forest".

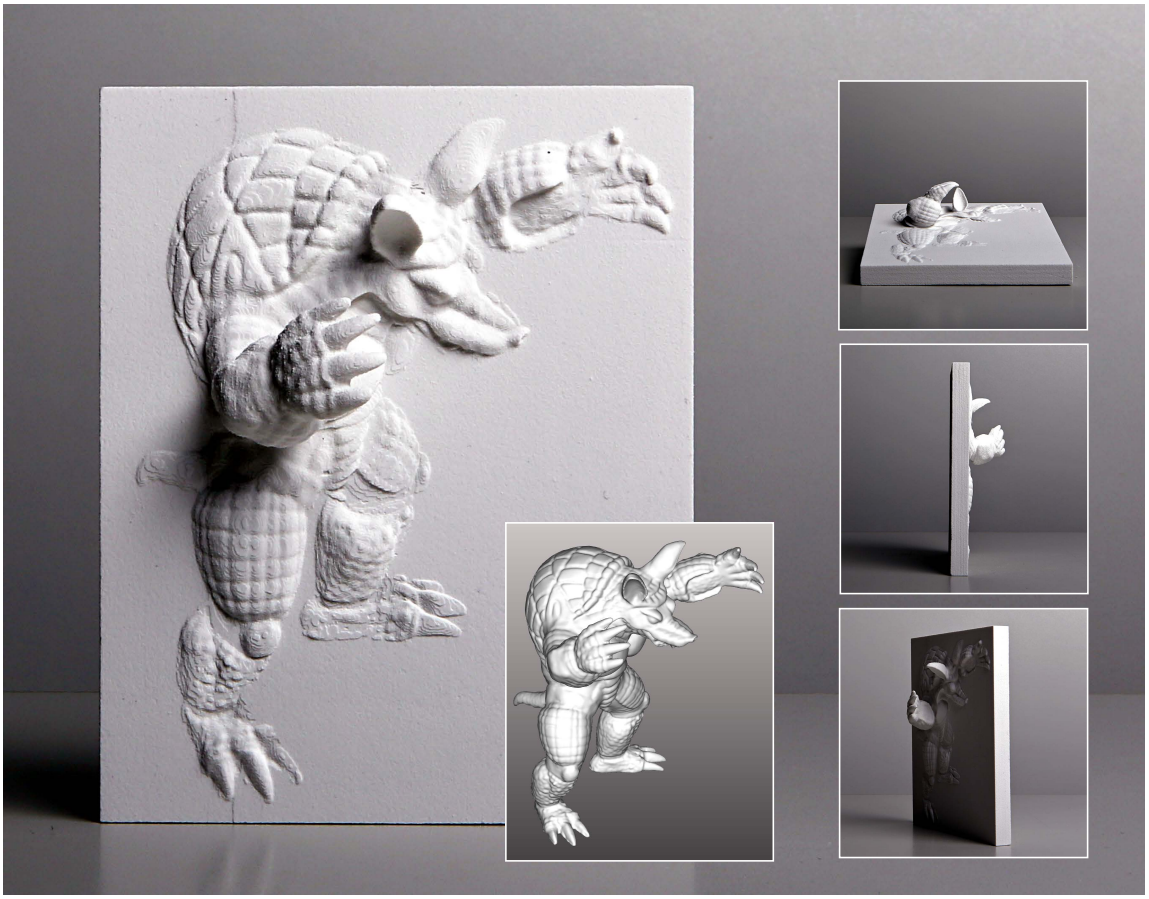


Figure 3.10: *A non-height field* Armadillo *relief, where the arm and one ear are left unconstrained, creating an interesting effect when observed from different angles.*

Inspired by street artwork painted over the steps of a staircase, we use our algorithm to embed a 3D model of an owl into a staircase, as shown in Figure 3.2. Note that the AMS is very thin, so it does not affect the function of the staircase, while being much more resistant than paint to erosion and aging. This idea is very general, and we plan to investigate it further in future works, applying it to design furniture, jewelry and architecture.

Stress tests. We stress test our method with two difficult sets of constraints. In Figure 3.12, we constrain the *Fish* model to stay within a 4 mm thin layer of a wavy surface mimicking the sea. The constraints strongly restrict the deformation, but our algorithm is still able to use the available space to generate a high-quality relief. The result is surprisingly close to the original model from the desired viewpoint, but the illusion immediately dissipates when the model is rotated.

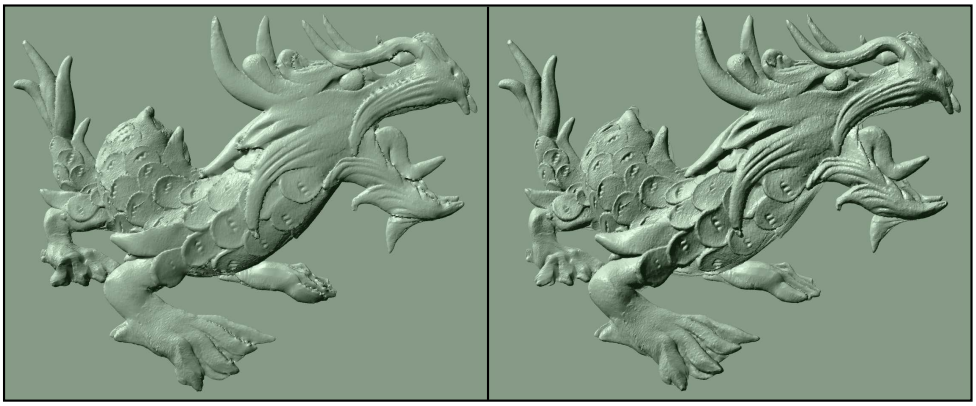


Figure 3.11: *Comparison of a Dragon bas-relief with [WDB+07].*

Carving bas-reliefs. Automatically generated bas-reliefs have mostly been created on flat or simple surfaces by previous algorithms. Our method can handle very complex cases without any modification, as we show in Figure 3.3. We carve a *Dragon Head* inside a V-shaped volume, constraining the surface to carve up to 6 mm in the volume and stick outside for no more than 1 mm. Both bounds are hard constraints that are guaranteed to be satisfied by our optimization.

Multiple view strips. Multiple views of the same *Head* model embedded in five planar surfaces are shown in Figure 3.15. Note that a highly nontrivial deformation is introduced to constrain the thickness of the different views to not exceed 2 mm.

Multiple objects. It is possible to create an AMS from a 3D scene composed of multiple disconnected objects. We show an example in Figure 3.18, where the visual hull of every cow is constrained to lie on the baseplate.

Natural lighting conditions. We demonstrate that our simplified lighting model (Section 3.2) is sufficient for the generation of realistic AMS in Figure 3.16, where we took photographs of our 3D printed results under outdoor lighting conditions. The photograph captured during a cloudy day (bottom) perfectly preserves the depth illusion, while direct exposure to sunlight (top) creates harsh shadows that reveal the extreme thinness of the *Fish* and the *Armadillo* reliefs.

Figure 3.12: *A fish embedded in the wavy surface of the sea, constrained to stay within a 4 mm thin layer.*

3.4 Concluding remarks

We presented a novel approach to create appearance-mimicking surfaces, which uses a special mesh parameterization of the deformation to robustly optimize for a surface whose normals are similar to the input geometry from a fixed perspective. Our algorithm supports exact and adaptive depth constraints and works directly on manifold triangle meshes without the need for resampling. We demonstrate the effectiveness of our approach to generate bas-reliefs and artistic compositions. Our lighting model assumes diffuse material and directional lighting and it does not account for self-shadowing. If these conditions are far from being satisfied, it produces sub-optimal results: We show a failure case in Figure 3.17, where a flash gun is placed close to the model and pointed directly at it. The harsh lighting creates specular

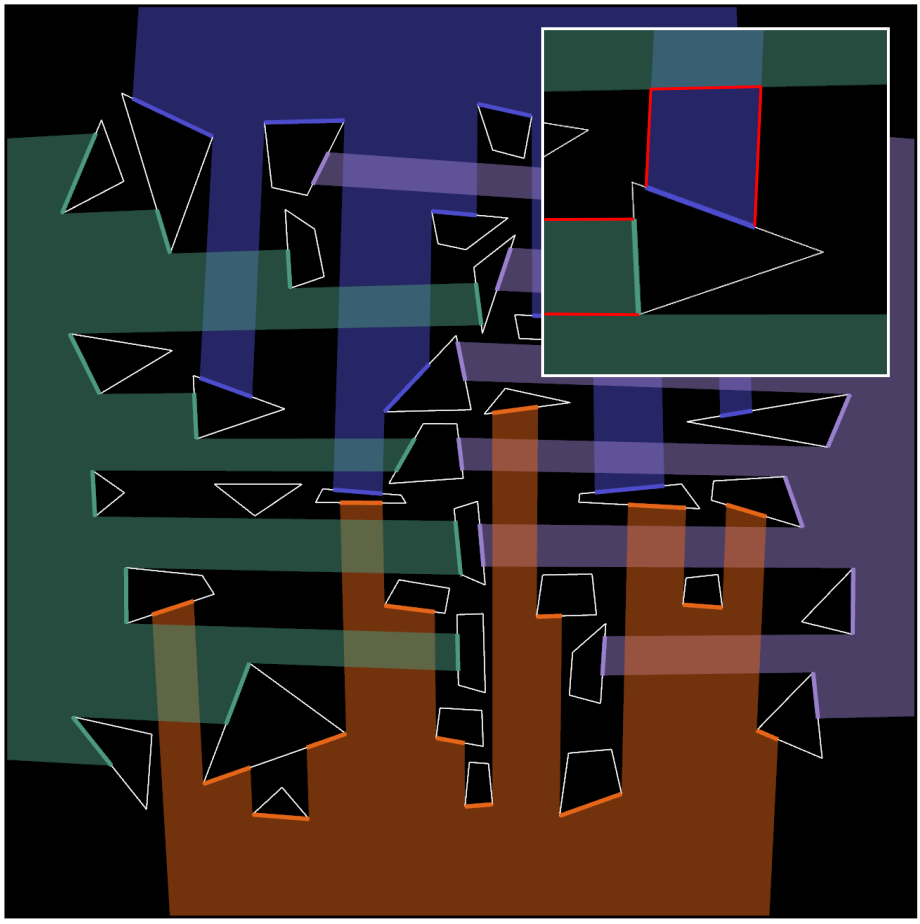


Figure 3.13: *Top down visualization of the "pillar forest" model (Figure 3.14) in our visualizer. Each color corresponds to the visibility range of a viewpoint. The inset shows the constrained spaces (red lines) for the fragmented model parts.*

reflections and strong shadows which ruin the illusion. We plan to investigate the use of a more accurate lighting model in future work.

The illusion generated by our method degrades as the viewpoint gets closer to the object, since the stereo disparity increases, helping the human vision system to detect the illusion. This problem has not been explored in the literature, and, given the quick development of 3D displays, is an interesting venue for future work.

With the advent of commodity 3D printing, we expect that bas-reliefs will be widely used to personalize objects and wearables. Our algorithm provides a robust and efficient way to support non-expert users in effectively using bas-reliefs in their creations.

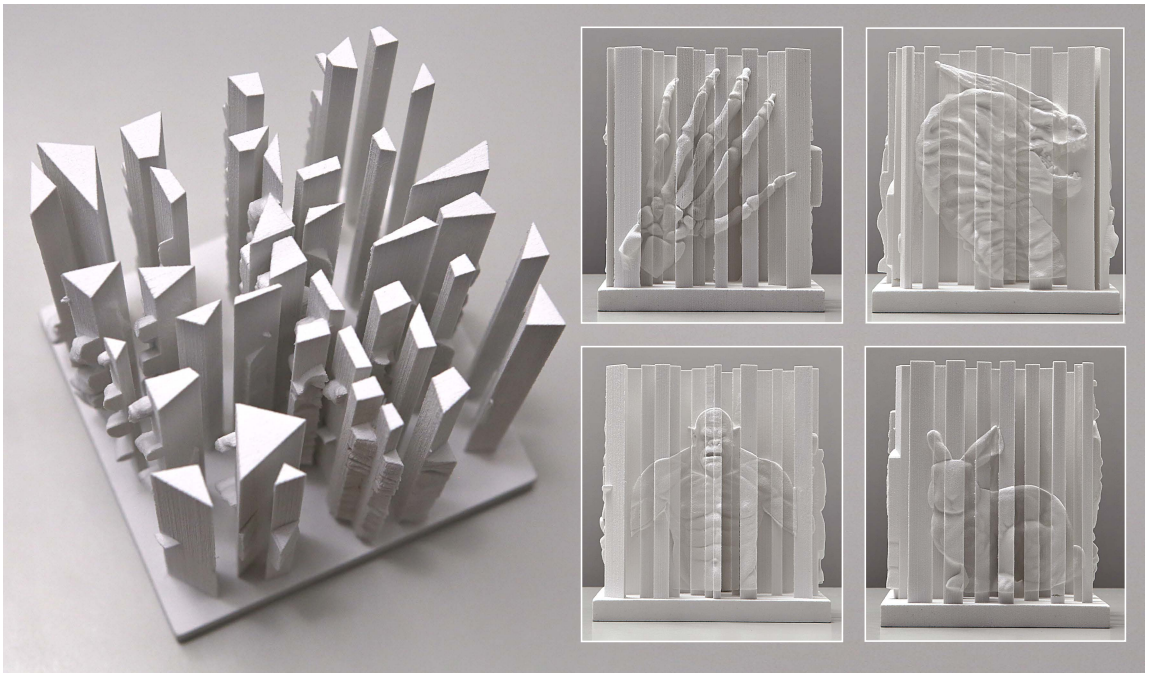


Figure 3.14: *Four 3D models are obfuscated in a "pillar forest".*

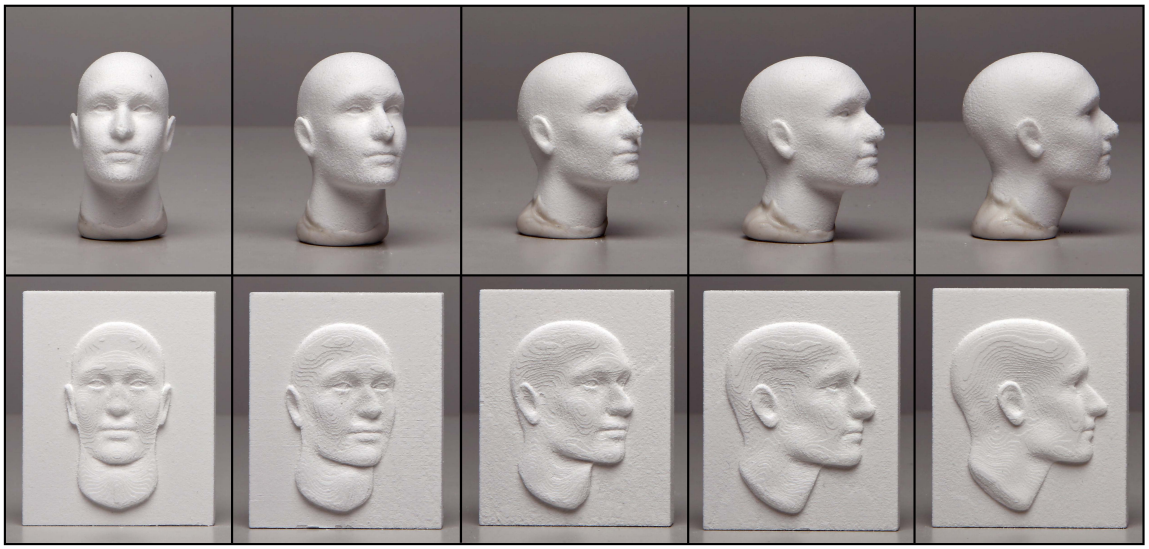


Figure 3.15: *Multiple views of the* Head *model (80k vertices), rotated by 22.5 degrees and constrained to a thickness of 2 mm.*

Figure 3.16: *A collection of appearance-mimicking surfaces photographed in a courtyard during a sunny day (top) and a cloudy day (bottom).*

Figure 3.17: *Under harsh lighting conditions the depth illusion does not work. This photograph has been taken with one single off-camera strobe casting light from left to right.*

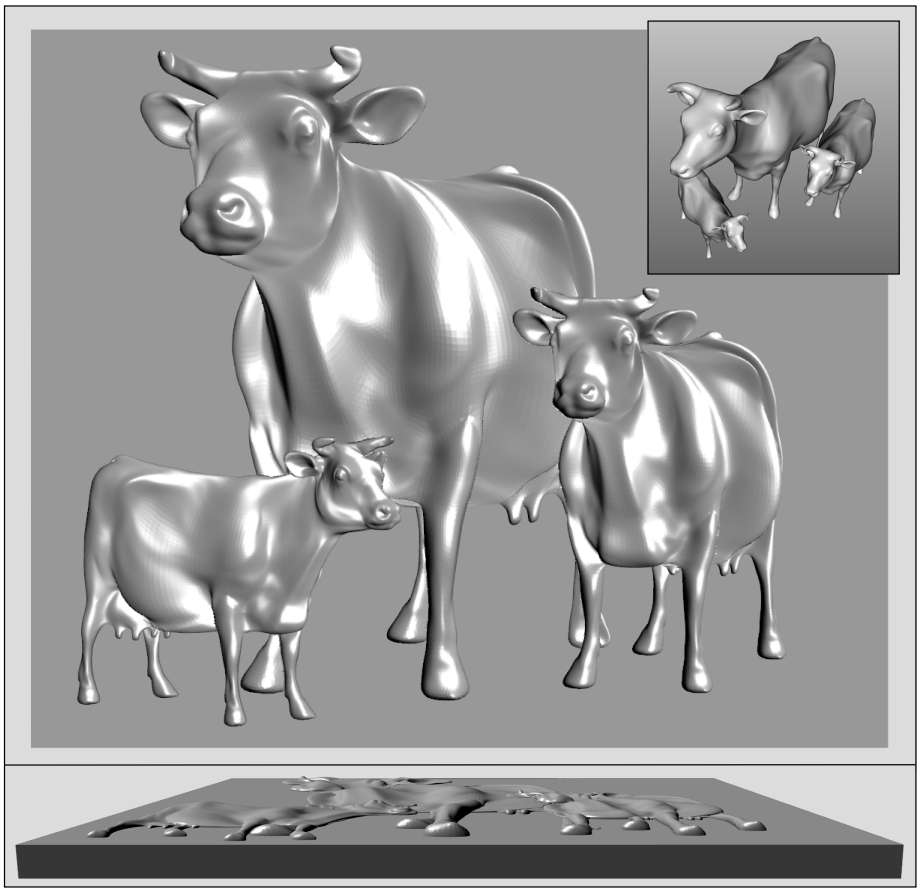


Figure 3.18: *A 3D scene containing a small cow family is converted to a thin relief. The inset shows a rendering of the original scene from the top view.*

CHAPTER 4

Replication of 3D colored objects with thermoforming

4.1 Overview

Automatically creating faithful physical replicas of digital 3D models is one of the major challenges in digital fabrication. Many fabrication techniques have been proposed to accurately reproduce the geometry of a 3D model, but very few methods can produce objects with a colored surface.

We propose a hardware and software solution to produce highly detailed textured objects using thermoforming [Kle09]. This can be done by printing onto the plastic sheet prior to deformation with a mold. This method has been limited to industrial applications so far, since printing on a thick plastic sheet requires a flatbed printer and heat-resistant inks, and the cost and effort of producing a mold is usually justifiable only if it is used to produce a considerable amount of thermoformed objects with the same shape.

Our proposal uses the same principle, but it is tailored to small-scale production and is accessible to universities, fabrication labs and hobbyists. On the algorithmic side, we propose a software simulation that creates the necessary pre-distorted texture image to be printed on the plastic, thereby ensuring that once the sheet is deformed, each pixel of the texture lands in its correct location on the 3D shape. The material model and the parameters for the simulation are automatically extracted by scanning and analyzing a single calibration object made with our forming pipeline. On the hardware side, we

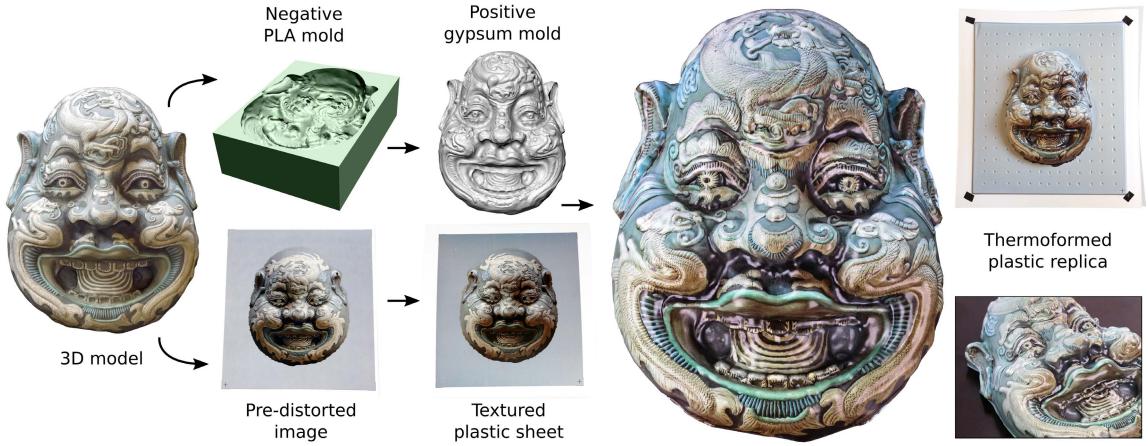


Figure 4.1: *Our pipeline for producing plastic replicas of textured digital 3D models by thermoforming.*

propose an effective method to produce a gypsum mold using a 3D printer with polylactic acid (PLA) filament, and a simple way to print texture on a plastic sheet using a standard color laser printer and transfer paper. The individual hardware components in our pipeline can be easily substituted thanks to our simple calibration procedure.

We validate our method with objective experiments that densely measure the fabrication errors, and with qualitative examples that demonstrate the variety of objects that can be fabricated with our technique. We provide comparisons with textured objects produced by hydrographic transfer and color 3D printing, where our technique provides superior quality while being considerably cheaper and faster.

We expect our contribution to have a strong impact both in digital fabrication, where it allows inexpensive production of highly detailed physical replicas of digital objects, and in industrial applications, where multiple designs can be easily tested before starting the mass production of thermoformed products.

4.2 Method

In this work, we use inexpensive off-the-shelf hardware and we demonstrate that it is sufficient to produce high-quality thermoformed objects. We start by detailing our hardware procedure, which combines thermal color transfer with thermoforming; we then explain the algorithm to generate the distorted texture image to be printed on the plastic sheet.

4.2.1 Hardware setup

Thermoforming is a manufacturing process where a plastic sheet is heated to a forming temperature, deformed to a specific shape in a mold, and trimmed to create a usable product. The procedure for the small manual thermoforming machines, which are commonly available in any fabrication lab, is divided into four steps:

1. *Preparation*: the mold is anchored to the vertically movable platform and lowered into the chamber of the forming machine. The plastic sheet is placed on top of the chamber to seal it and is clamped by a metal frame.
2. *Heating*: the plastic sheet is uniformly heated to forming temperature, which is above the glass transition of the plastic.
3. *Mold raise*: the mold is raised and pushed into the plastic sheet.
4. *Vacuum*: a vacuum is created between the mold and the sheet, resulting in forces that pull the heated plastic to the mold.

The deformed plastic is optionally trimmed to remove the border. The above technique can produce colored objects by printing an image onto the plastic sheet before thermoforming it. The printing requires the use of dedicated, expensive flatbed-printers and a special heat resistant ink [FUJ16] to withstand the high temperature and stretch of the surface in the vacuum forming process, limiting it's applicability to an industrial setting.

Texture transfer. We found that with a special thermal transfer paper [tra15], it is possible to inexpensively print and thermoform high-resolution images on plastic sheets with a visual quality comparable to the industrial approach. The procedure is simple and similar to the printing of custom graphics on T-shirts. First an image is printed on the transfer paper using a standard office laser printer. Then it is transfered onto plastic with a common thermal press, gluing the toner particles onto the sheet's surface. The resulting prints have a vivid color and are robust to heat and stretch deformation in the thermoforming process (Figure 4.2).

Mold fabrication. We use a combination of 3D printing and casting to fabricate heat-resistant molds. Using boolean operations, we produce a negative copy of the 3D shape we would like to replicate (Figure 4.3). The negative mold is then 3D printed using a PLA printer and filled with a gypsum water mix. The extraction of the cast would be extremely difficult for complex shapes with concavities, but, incidentally, PLA is the perfect

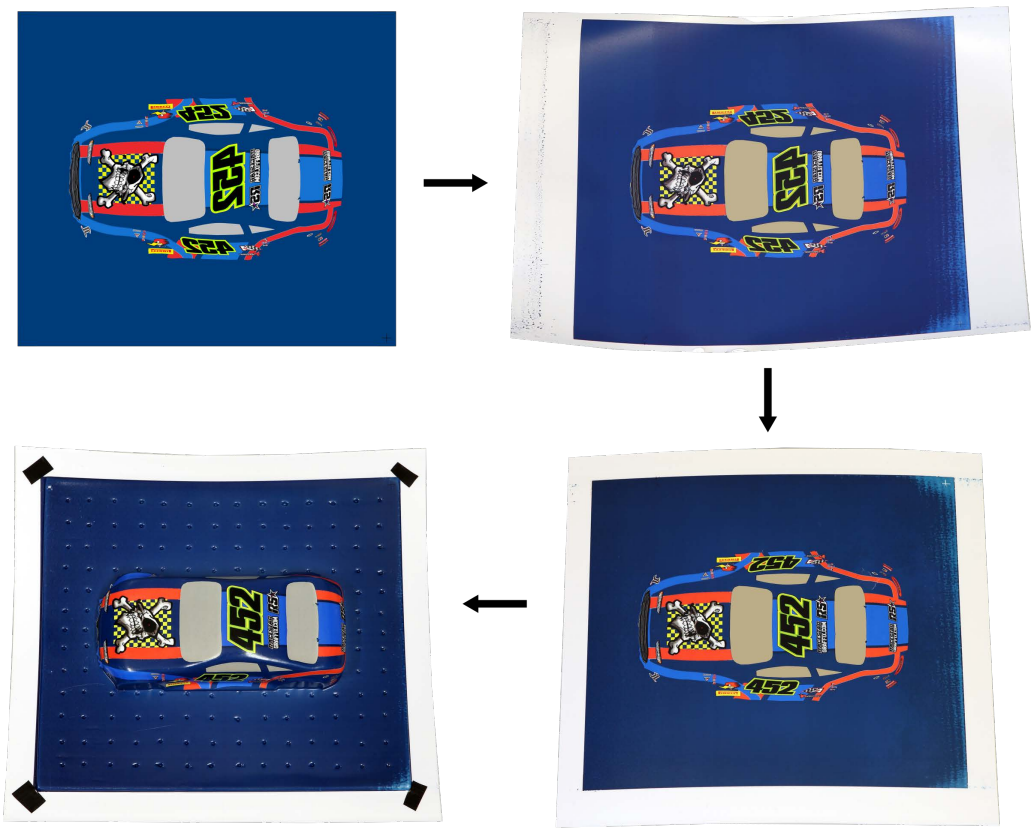


Figure 4.2: *Original image, printed image, transferred on plastic, thermoformed.*

material for this application. We can heat the PLA using a heat gun until it melts and then extract the gypsum mold without any risk of damaging it (Figure 4.3).

Off-the-shelf hardware. This novel combination of texture transfer and mold fabrication drastically reduces the cost and the time needed to thermoform colored objects without compromising the quality of the results. We used this technique to produce all presented results.

4.2.2 Simulation

The thermoforming process induces a complex deformation of the plastic sheet to adapt its geometry to the mold. We propose an algorithm to simulate this deformation and to invert it, effectively converting a textured digital 3D model into the 2D image to be printed on the plastic sheet prior to deformation.

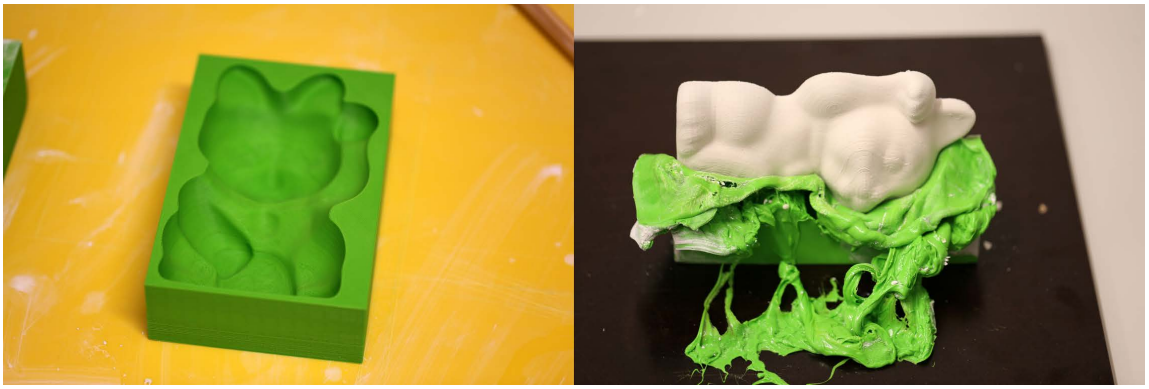


Figure 4.3: *Negative mold, half-melted mold revealing the positive gypsum cast.*

Assumptions. To simplify the simulation, we make the following assumptions:

1. The adhesion force between the plastic and the mold is infinite [NTD90]: the parts of the sheet that touch the mold move rigidly with it.
2. The temperature of the plastic sheet is uniform, and the thermoforming process is fast, making the effect of the plastic cooling negligible.
3. The change in pressure due to the activation of the vacuum pump is instantaneous.

These assumptions allow us to design an efficient algorithm for the entire simulation, enabling to find the optimal parameters for a specific hardware setup using a grid search (Section 4.2.3). The efficiency of our method is also convenient while preparing new designs: it only takes us minutes to compute and visualize a digital preview of the thermoforming result of a model with complex geometry.

We use the thin sheet model proposed in [BUAG12] with an additional plasticity model and a simplified handling of contacts. For the sake of completeness, in the following we report all details of our simulation.

Discrete viscous sheets [BUAG12]. The plastic sheet (membrane) is represented as a triangle mesh with a scalar field on faces, whose values represent the thickness of the sheet. To denote quantities in the undeformed (reference) state, we use a bar over the respective letter. To model the stretching of the membrane we use the hyperelastic St. Venant-Kirchhoff material model. For

a single triangle the Green strain is defined as

$$\varepsilon = \frac{1}{16A^2} \sum_{i=1}^{3} (l_i^2 - \bar{l}_i^2)(t_j \otimes t_k + t_k \otimes t_j), \tag{4.1}$$

where A is the triangle area, l_i and $\bar{l}_i$ are deformed and reference lengths of edge i, respectively. $\otimes$ denotes the outer product of two vectors, and t_i is the outward normal to the edge i in the plane of the undeformed triangle. The strain-energy potential over the surface is:

$$E_s = \frac{1}{2} \sum_t A_t \frac{Y h_t}{(1-v^2)} \left((1-v) Tr\left((\varepsilon_t)^2 \right) + v (Tr(\varepsilon_t))^2 \right),$$

where Y is Young's modulus, v is Poisson's ratio, and h_t is the thickness value of triangle t.

The membrane model does not capture the bending of the membrane, which is modeled separately by a term that depends on the dihedral angle θ_e between each pair of faces sharing an edge e:

$$E_b = \sum_e \frac{Y h_e^3}{12(1-v^2)} \frac{3\bar{l}_e^2}{A_s} (\theta_e - \bar{\theta}_e)^2,$$

where h_e is the mean thickness of the two incident triangles on e and A_s their summed area.

Viscosity is modeled using forces derived from a discrete dissipative potential:

$$E_{\text{visc}} = (o / \triangle t)\, E(p_{k+1}, p_k),$$

where E is an elastic potential expressed in terms of the deformed and a reference material configuration. In this case these are the configurations between two consecutive integration steps p_{k+1} and p_k, where $\triangle t$ is the step size and the scalar o controls the viscosity. For the stretching energy we have $E_s(\varepsilon_{k+1}, \varepsilon_k)$, where ε_{k+1} and ε_k are the corresponding strains. Equivalently, for the bending energy we use $E_b(\theta_{e,k+1}, \theta_{e,k})$. Viscous forces are then computed by differentiation of E_{visc} with respect to the end-of-step configuration p_{k+1}.

Incompressibility is enforced by updating the thickness values h of all triangles after each integration step to be $h = V/A$, where A is the current triangle area and V it's constant volume.

The vertex positions are updated at each step using a first order implicit Euler (backward Euler) integration; we refer to [WB97] for a detailed description. This concludes the summary of the model proposed in [BUAG12]. In the following we discuss the extensions necessary to adapt it to simulate a thermoforming process.

Plasticity. We experimented with the fully viscous model proposed in [BUAG12] and realized that it is problematic during the heating phase of the thermoforming process: with pure viscosity, the sheet flows down the forming chamber instead of only slightly bending. We found that the behavior of heated plastic can be approximated well with an additive model [Hil98, SH98]:

$$\varepsilon = \varepsilon^e + \varepsilon^p,$$

where the strain ε is divided into elastic and plastic parts. Following [MG04, OBH02], we similarly update the plastic strain after every time step only if its norm exceeds the yield strain c_{yield} (which is a property of the material):

$$\text{if } \|\varepsilon\|_2 > c_{\text{yield}},\ \ \varepsilon^p_{k+1} = \varepsilon^p_k + \triangle t \cdot c_{\text{creep}} \cdot \varepsilon^e,$$

where c_{creep} is a material parameter that controls the plastic flow velocity. Note that the update of the elastic strain cannot be bigger than $\triangle t \cdot c_{\text{creep}} \leq 1$.

Plastic rest pose. We update the rest pose of the mesh to directly account for the plastic deformation of the sheet.

By fixing an orthogonal reference system for each triangle whose third axis is parallel to the triangle normal, the symmetric Green strain tensor (Eq. (4.1)) has the following form:

$$\varepsilon = \begin{bmatrix} e_1 & e_2 & 0 \\ e_2 & e_3 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where there are only 3 independent coefficients. Note that in Eq. (4.1), the strain is expressed as a linear combination of the difference of the squared edge lengths $s_i = l_i^2 - \bar{l}_i^2$. We can thus rewrite this relation in matrix form as:

$$\mathbf{Ts} = \mathbf{e},$$

where $\mathbf{s} = [s_1, s_2, s_3]$ and $\mathbf{e} = [e_1, e_2, e_3]$. Solving this linear system (and removing the constant $\bar{l}_i^2$) gives us the squared edge lengths of the rest pose. To reconstruct the mesh, we consider each triangle independently, and we reconstruct its geometric embedding using the closed-form formula in the Appendix.

Since the membrane is very thin, we simplify the bending plasticity formulation by always updating the rest pose dihedral angles with the current ones, following [BUAG12].

External forces. There are two kinds of external forces that act on the mesh vertices: The first is gravity, which is modeled as a constant force f_g in the negative z direction. The vacuum pressure induces forces proportional to the vertex Voronoi areas, oriented in the opposite direction of the surface normal. Gravity is active during the entire simulation, while the vacuum acts only in the last stage.

Contacts. Since the adhesion forces between the plastic and the mold are dominant [NTD90], we glue the sheet to the mold upon contact. We only check for collisions between the sheet vertices and mold triangles and use hard positional constraints to move them rigidly with the mold. To account for the thickness of the sheet, we interpolate the triangle thickness values to the vertices and perform the collision detection on two offset surfaces, using EMBREE [WWB+14] to check for vertex-triangle collisions.

Simulation stages. Equipped with this simulation model, we run our simulation in three phases, illustrated in Figure 4.4:

1. *Relaxation* (1 second). In this phase, the plastic sheet is fixed on its border and the only external force is gravity.
2. *Raising the mold* (about 1 second). The mold is raised with a speed of $0.1\,\mathrm{m/s}$. During this phase, the mold touches the sheet, raising and deforming it.
3. *Vacuum* (until convergence). The vacuum forces are activated and the sheet is pulled toward the mold.

The simulation is interrupted when all vertices touch either the mold or the base plate or when the simulation time exceeds 5 seconds.

4.2.3 Calibration

Different types of plastic and thermoforming hardware setups produce different results, requiring different simulation parameters to accurately model their behavior. Classically, these parameters are computed from the specifications of the thermoforming hardware (temperature, vacuum pressure, speed of the moving platform, etc.) and the material used, and are acquired via physical material tests. To avoid this difficult and error prone procedure and enable the usage of various hardware setups, we propose an automatic procedure that relies only on fabricating and 3D scanning a single calibration object.

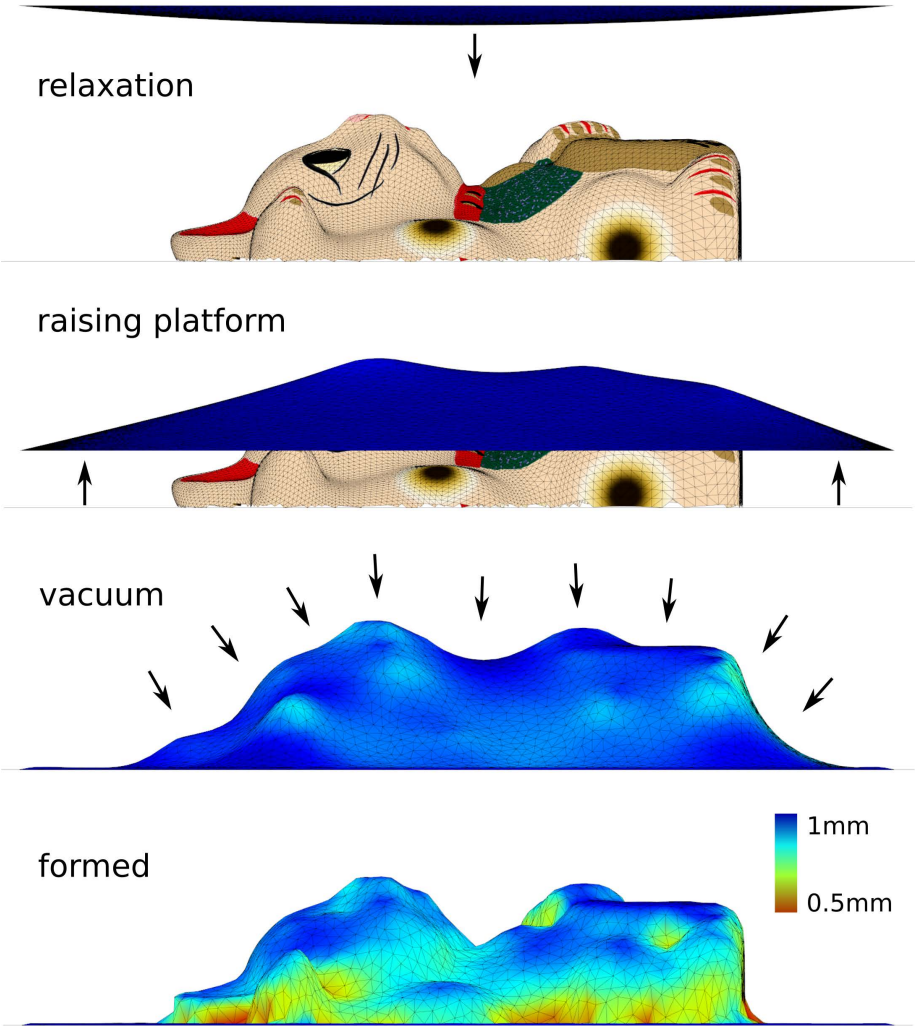
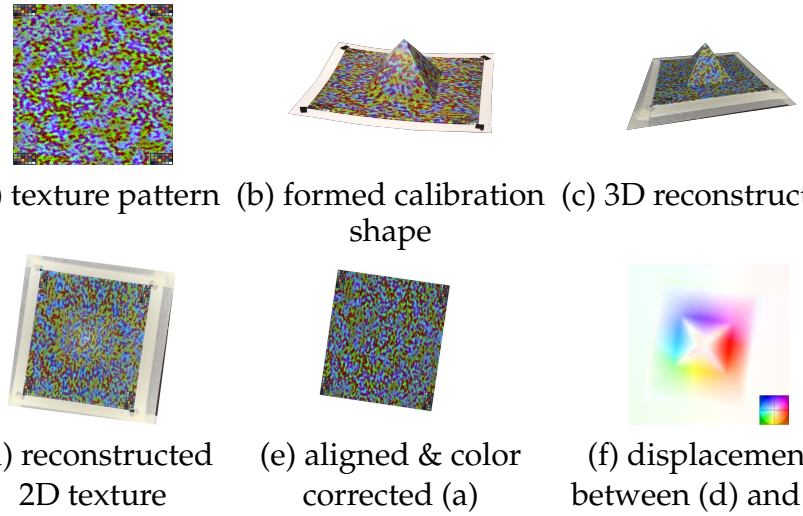


Figure 4.4: *The simulation is divided in three phases: from top to bottom, relaxation, raising the mold, activating the vacuum. The color of the sheet visualizes the thickness of the simulated plastic sheet.*

Calibration object and texture pattern. Our calibration object is a pyramid, chosen because it is simple to fabricate and scan (Figure 4.5). Other shapes could also be used, the only requirement being that they should be easy to 3D scan after the thermoforming. As we measure the thermoforming deformations from a 3D reconstruction of the textured object, we designed an RGB texture pattern (Figure 4.5 (a)) that has dense features that are easy to automatically detect. The pattern is then transferred to the plastic sheet and thermoformed (Figure 4.5 (b)). We obtain this pattern by computing Gaussian noise of different resolutions in each of the RGB color channels. This results in an image with well distinguishable features at various frequencies distributed over the RGB color channels, allowing us to reliably detect deformations of



(a) texture pattern (b) formed calibration shape (c) 3D reconstruction

(d) reconstructed 2D texture (e) aligned & color corrected (a) (f) displacements between (d) and (e)

Figure 4.5: *Calibration process. We print a specifically designed texture pattern (a) on a plastic sheet and perform thermoforming with a calibration shape (b). The result is 3D scanned (c) and a 2D texture is computed (d). After alignment and color correction of the texture pattern (e) we estimate the displacements in the material (f). The inset in (f) illustrates the color coding of the displacement vectors.*

different magnitudes. This pattern can be seen as an extension of the greyscale wavelet pattern of [AIH+08]. Additionally, we print standard color checker charts [XR16] on the pattern to enable color correction of the captured images to compensate for imperfections in the camera hardware and capture setup.

3D reconstruction. We capture about 100 high-resolution images of the thermoformed calibration object using a Canon 6D camera and feed them into an off-the-shelf multi-view reconstruction system [Agi16] to compute a 3D model of the object (Figure 4.5 (c)). To increase the quality and robustness of the deformation estimation under uneven lighting conditions, we also perform color correction on the texture using the color checkers embedded in the pattern.

Preprocessing. The reconstructed 3D model is then flattened onto the UV domain using the as-rigid-as-possible parameterization algorithm [LZX+08] which results in the 2D texture shown in (Figure 4.5 (d)). The input texture pattern (Figure 4.5 (a)) is roughly aligned to it using a homography transformation, which maps the 4 corners of the input pattern to the 4 corners of the parameterized model (manually selected). An aligned and color correction input pattern is shown in (Figure 4.5 (e)).

Deformation estimation. After preprocessing, we estimate the deformation by computing a dense displacement field between the reconstructed 2D texture and the input texture. We use a well-established optical flow method [BBPW04] for computing the displacement field, which can, thanks to our special pattern, robustly reconstruct the flow even for large and complex deformations. To speed up the computation, we run the optical flow solver on images of size 2K×2K pixels, which we found to be sufficient to obtain an accurate deformation estimation. Computing a single flow field at this resolution takes about 32 s on a single workstation (Intel Xeon E5-1680 v3, 64 GB RAM) using a CPU implementation. We use a fixed set of parameters for all results: smoothness weight $\alpha = 20$, gradient weight $\gamma = 5$ and pyramid steepness $\eta = 0.95$. Please refer to the original paper [BBPW04] for an explanation of these parameters. The final deformation is estimated from the optical flow field by applying the inverse mapping of the UV coordinates and the homography transformation used for the initial pattern alignment.

While it might be tempting to directly use this dense map to compute the deformed pattern to print, entirely sidestepping the need for the simulation and parameter fitting, this is only possible for simple geometries that can be easily scanned with high accuracy. Besides the additional production costs, it is also much more time-consuming to fabricate and scan the object with the calibration texture than to run our simulation, which only takes 5 minutes.

Parameters and grid search. Our simulation depends on the following parameters: Young's modulus (Y), creep (c_{creep}), yield strain (c_{yield}), viscosity (v), Poisson's ratio, dimensions and density of the sheet, vacuum pressure and elevation speed of the mold. Since our model is a simplified approximation of the true physical vacuum forming process, the parameters from material tables do not necessarily minimize the alignment error. Therefore, we optimize for them using material tables to define reasonable ranges.

We experimentally observed that only the first 4 parameters need to be optimized to obtain an accurate simulation, while the others can be copied from a material table (Poisson's ratio: 0.35, sheet density: $1330\,\mathrm{kg/m^3}$), easily measured like the thickness (1 mm) and size ($24 \times 26\,\mathrm{cm}^2$) of the sheet and elevation speed ($0.1\,\mathrm{m/s}$), or found in a specification sheet (vacuum pressure: 80 kPa).

We restrict these four parameters to lie in plausible ranges ($Y \in [5 \cdot 10^5, 5 \cdot 10^6], c_{\text{creep}} \in [0, 1000], c_{\text{yield}} \in [0, 0.1], v \in [0, 10^{-4}]$) and we search in this restricted space using a grid-search approach. We sample 625 points (5 per dimension) and pick the ones with the lowest average error with respect

to the ground truth. The error is measured as the average of the Euclidean distance between our simulation and the acquired ground truth. To account for registration errors, we optimize for a small translation (up to 2 mm) in the mold by uniformly sampling the space of translations and picking the best candidate. The parameters we found with this procedure for our hardware setup are $Y = 2.75 \cdot 10^6$, $c_{\text{creep}} = 500$, $c_{\text{yield}} = 0.1$ and $v = 2.5 \cdot 10^{-5}$.

4.2.4 Computational thermoforming

After introducing the hardware, the simulation and the calibration details, we now present our complete thermoforming pipeline, which converts a textured digital 3D model into a high-resolution plastic replica (Figure 4.1).

1. *Simulation.* The 3D model is used as the mold in the simulation (Section 4.2.2), which is run using the parameters obtained from the automatic calibration procedure (Section 4.2.3). The simulation starts with a flat triangle mesh model $M = \{V, F\}$ of the plastic sheet and produces a new set of vertex positions V' that correspond to the sheet after thermoforming.
2. *Projection.* After simulation, the input 3D model is projected onto the simulated mesh of the plastic sheet $M' = \{V', F\}$ using ray casting. For each vertex of the 3D model, we compute its barycentric coordinates in M' and then use the same coordinates to find the corresponding point in M. The image to print on the plastic is obtained by rendering the now flattened 3D model, using these new locations in M.
3. *Mold creation.* The 3D model is subtracted (in the Boolean sense) from a box to create a negative model of the mold. The negative is 3D printed and used to fabricate the gypsum mold (Section 4.2.1).
4. *Texture transfer.* The image is printed and transferred onto a plastic sheet using thermic transfer paper (Section 4.2.1).
5. *Thermoforming.* The plastic sheet is thermoformed, producing the textured replica.

4.3 Results

We ran our simulation algorithm on a dual processor workstation (Xeon CPU E5-2650 v2, 64 GB RAM) and used PARDISO [SWH07, SBR08, KLS13] to solve the involved linear systems. We discretize the plastic sheet using a mesh with 10K triangles. The time needed for the simulation is mostly

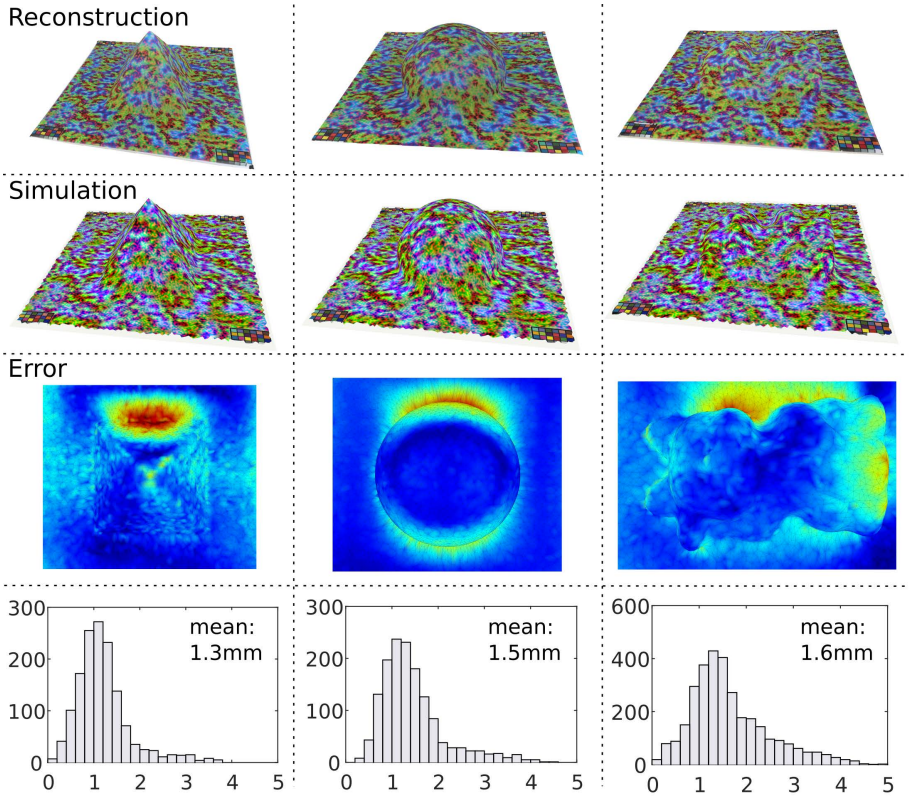


Figure 4.6: *Left, from top to bottom: the 3D reconstruction of the calibration shape, our simulation result, the visualization of the Euclidean distance error between the reconstruction and our simulation (dark blue: small error, dark red: higher error). The corresponding histogram in the bottom shows the distribution of alignment errors in millimeters. Note that this shape has been used for calibration and therefore has the smallest error. In the middle and on the left, we show our validation, computing the error on a hemisphere and on the cat shape using the same parameters. The error distributions in the histograms are very similar, suggesting that our simulation accurately reproduces the thermoforming process.*

independent of the geometry used and is around 5 minutes. The offline calibration procedure takes about 40 hours, mostly not involving any user interaction: 1 hour to fabricate the calibration object, 30 minutes to take the photographs, 2 hours for 3D reconstruction, 1 minute for optical flow and 36 hours for the parameter grid search.

Quantitative evaluation. We thermoformed a hemisphere and the cat model and used our calibration pipeline to measure the deformation introduced by the thermoforming. We then compared it with the result of our simulation, using the parameters we previously estimated on our pyramid calibration object. We obtained an average displacement error of 1.5 mm and

Figure 4.7: *Our replica of the cat model (right) has a superior surface quality to that created with a ZCorp 650 powder printer [3ds16] (left) or hydrographic transfer [PDP+15] (middle).*

1.6 mm, respectively, which is close to the best fitting error that we got on the pyramid for the calibration (1.3 mm). A visual comparison of the errors and their corresponding histograms are shown in Figure 4.6. The distribution of the errors shown in the third row suggests that there is some non-uniformity in the actual thermoforming process. A possible source could be the imperfection of the heating system, which we assumed to be uniform and did not include in our model.

Comparisons. We show a comparison between our fabrication technique and two competing methods in Figure 4.7. Our result is not affected by the flat regions that cause artifacts in the hydrographics technique proposed in [PDP+15], and it has a superior resolution in respect to powder-based printing techniques.

Fabricated examples. We fabricated various models to test our method and potential applications (Figure 4.8). Mimicking the plastic food replicas commonly used by restaurants in Japan, we fabricated two loaves of bread of different sizes (Figure 4.9). Since the objects produced with our technique are lightweight and very robust, the method is ideally suited to produce scenery pieces for model building, such as a mountain miniature (Figure 4.11) or a stump (Figure 4.10). By thermoforming transparent plastic, it is possible to

Figure 4.8: *An overview of the examples produced with our method.*

obtain replicas of objects that contain transparent parts. In Figure 4.12, we fabricate the shell of a radio controlled car, leaving the windows transparent. This technique could be particularly useful for creating customized product packaging, as we demonstrate in Figure 4.13. Extremely detailed objects can also be fabricated with our technique, such as the Chinese mask in Figure 4.14. This object has many detailed features, which are accurately preserved in the physical replica.

Limitations and future work. The main limitation of our work lies in the registration between the mold and the printed plastic sheet before thermoforming. We currently use visual markers and perform the alignment by hand, which results in an alignment error of up to 2 mm. This could be avoided by using a customized thermoforming machine, but it would be an interesting challenge to tackle this problem using a lower cost approach that does not require special hardware. Currently, our solution only supports single-layered plastic sheets and cannot be used to produce closed objects. Depending on the complexity of the geometry, it might also be difficult to remove the gypsum mold.

An interesting avenue for future work would be the automatic design of

Figure 4.9: *Plastic food samples can be fabricated with our technique, avoiding hours of manual painting.*

decomposable molds, enabling thermoforming-based fabrication of objects with large concavities.

4.4 Concluding remarks

We proposed a new digital fabrication method to manufacture objects with a high resolution texture using thermoforming. Our solution relies on common hardware available in many digital fabrication labs and produces objects with a surface quality greatly superior to competing techniques. We believe computational thermoforming will have a significant impact in the fabrication community thanks to its low cost, low hardware requirements, high fabrication speed and quality, and that it has the potential to be a valuable tool for industries to quickly experiment with different thermoformed product designs.

Figure 4.10: *A replica of a miniature modeling stump. Note how the texture aligns with the model's geometric features.*

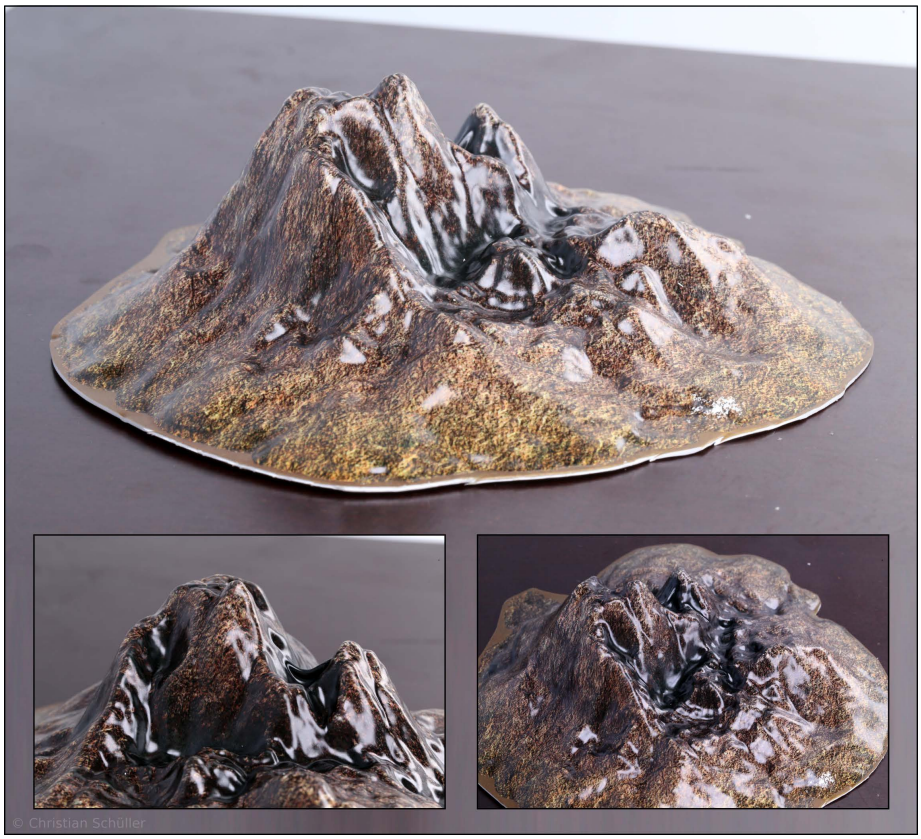


Figure 4.11: *A scaled replica of a mountain. The lightweight and robust material is well suited for application in model building.*

Figure 4.12: *An RC car shell fabricated with our method using transparent plastic.*

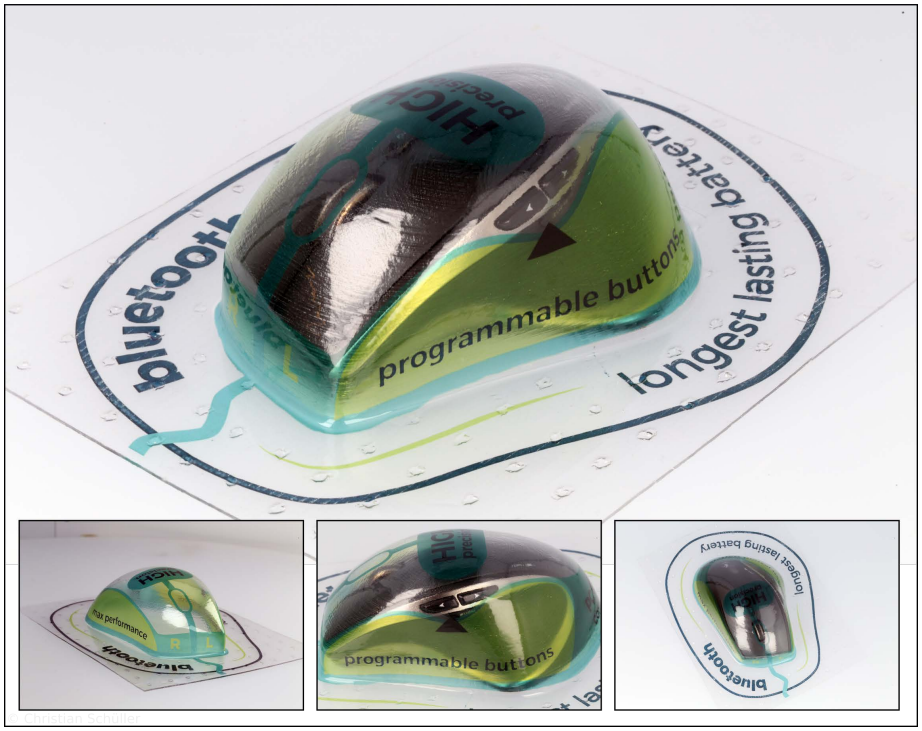


Figure 4.13: *A customized mouse packaging produced with our method.*

Figure 4.14: *Our technique can faithfully replicate extremely detailed models.*

CHAPTER 5

Design and fabrication of zippable 3D shapes

5.1 Overview

Representing shapes using developable surfaces is a problem with numerous applications, ranging from recreational activities like papercraft and plush toy fabrication, to large scale industrial design and modern architecture. We introduce the concept of *zippables* – two dimensional, branching, ribbon-like pieces of fabric that can be zipped up to form 3D shapes. Our interest in this problem is inspired by a product commercially known as the *zipit* bag [zip17]. This bag is made from a single, long piece of zipper. When zipped up, the ribbon folds and wraps around to form a simple bag. It takes only a few seconds to zip up and down, and no instructions are necessary. The simplicity of this concept immediately propelled us to ask whether this idea can be employed and generalized to facilitate the fast fabrication of *arbitrary* shapes. To this end, we devise a computational method to create a developable shape that approximates a given target 3D model when zipped up; see, e.g. Fig. 5.1. Our approach generalizes the simple straight ribbons that make up the *zipit* bags by allowing the zippables to turn, have varying width, and branch (see Fig. 5.2, bottom right inset).

We approach the problem in two stages. First, we compute a single curve on the surface, which represents the zipper-curve, i.e., the 3D path that the zipper should take. We then approximate the original surface geometry by a single developable surface – the zippable – whose boundary interpolates the zipper-

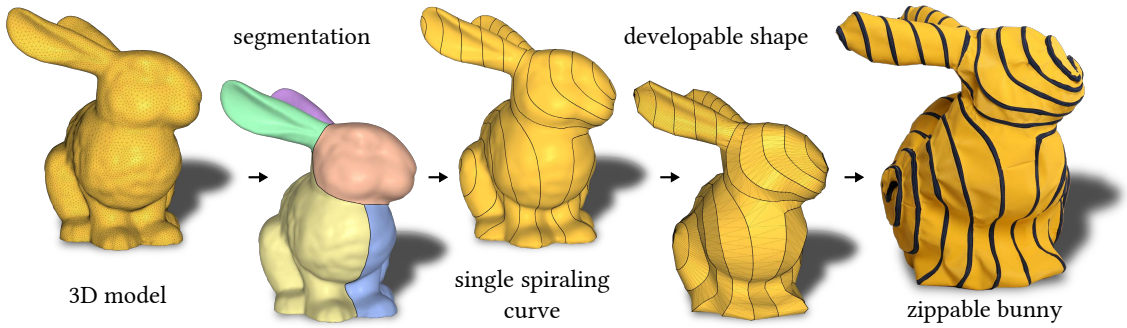


Figure 5.1: *The pipeline of our approach. Starting from a 3D model, the user decomposes the shape into topological cylinders. Our algorithm automatically produces a single continuous curve on the shape that spirals along the cylinders. It proceeds to cut the shape along the curve and creates a developable surface that can be trivially unfolded into a single 2D shape – the so called* zippable*. Based on the flattening, plans for laser cutting it from fabric are generated. Finally, we attach a zipper with a single slider to the boundary of the zippable. Zipping it up reproduces a faithful approximation of the input model.*

curve (see Figures 5.2 and 5.4). Since the resulting shape is largely determined by the first stage, several considerations must be taken into account when planning the zipper-curve: first, it should cover the surface as uniformly as possible, in order to get a uniform approximation. Equivalently, the zippable should have as little variation of width as possible. Second, the zipper-line should not curve excessively, as zippers tend to resist bending in the plane (see Fig. 5.17) and attaching them to a sharply turning curve is challenging.

There are several sensible strategies for tracing a zipper-curve. We discuss them and their limitations in Sec. 2.5.1. Our main observation is that the two aforementioned properties, uniformity and curvature, are trivial to achieve when the target 3D model is a cylinder: simply draw a spiraling curve on the cylinder from one boundary to the other, such that if cut along that curve, the resulting shape is a straight ribbon of constant width. For general target surfaces, our approach is based on first decomposing the shape into topological cylinders, and then mapping them onto cylindrical domains with low isometric distortion and in a seamless manner. We then draw "perfect" spirals on the cylinders and map these spirals back onto the input shape. Since the mappings minimize isometric distortion, the mapped curves tend to exhibit low curvature and low variation of distance between windings. Inspired by [ZGH+16], we connect the spirals on the different cylinders into a single, long curve using *Fermat spirals* (see inset). The shape is then cut along the computed curve to create a single,

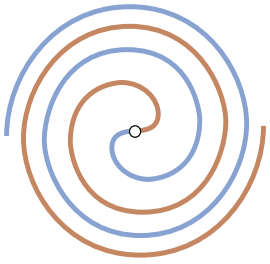


Figure 5.2: *The star model fabricated with our fastening rig. The insets on the left and in the middle show the developable model from a front and side perspective. Note how the fabricated star perfectly resembles the predicted shape. The segmentation is shown in the top right corner; below it a visualization of the zipper tape and the flattened* zippable*. The physical result has a height of 27 cm.*

possibly bifurcated, but not yet developable, ribbon-like surface. A simple remeshing process transforms this surface into a developable one, essentially making a zippable. It can be trivially unfolded onto the 2D plane to create a cutting pattern. The resulting strands of the flat ribbon might overlap in the plane, and the pattern might take up too much space to be cut with an available laser cutter; in both cases we simply divide the ribbon into a few separate pieces before cutting them from fabric and attaching a zipper.

In addition to the final zipping up being easy and entertaining, our assembly and fabrication process has distinct advantages over papercraft and many other similar methods in this domain. Most importantly, attaching the zipper to the zippable can be done by working solely in the flat plane. In contrast, attaching multiple parts in papercraft or sewing plush toys from multiple charts usually cannot be done in a flat configuration, but rather requires a certain assembly order and sewing in 3D, especially in the final stages, where all the parts have to come together for the shape to close up. The makers

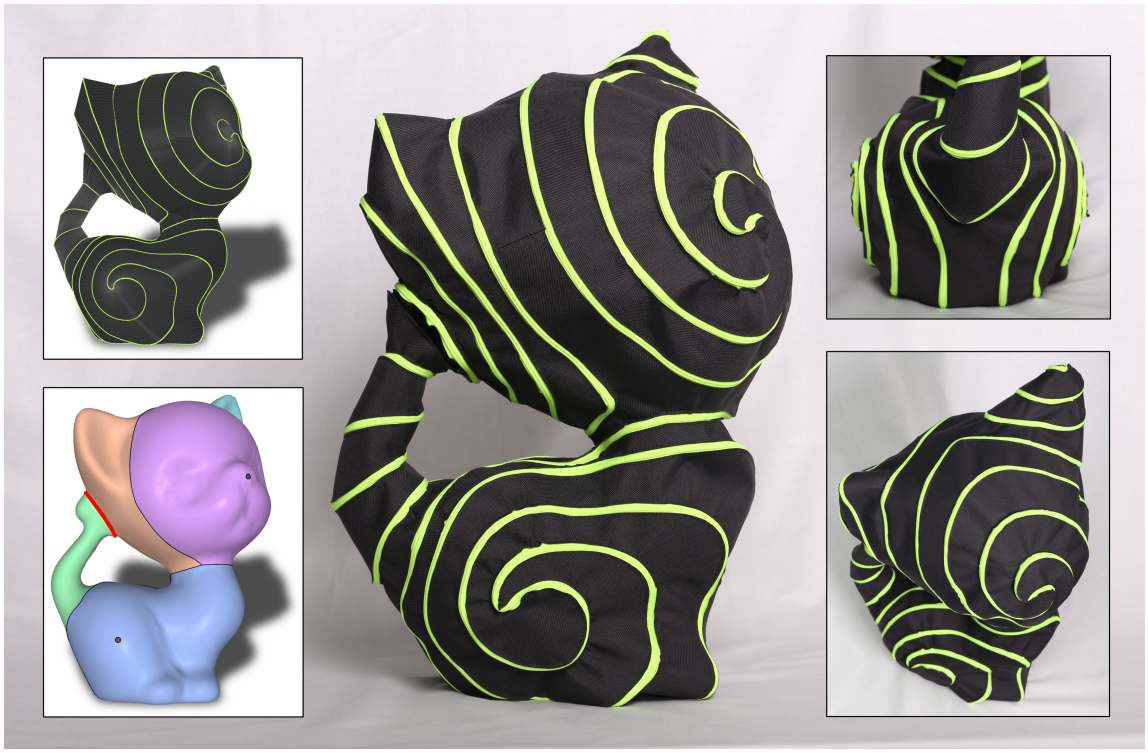


Figure 5.3: *A zippable shape of a kitten. Since it is topologically equivalent to a torus, an additional cut is needed (bottom left inset: marked in red where the tail touches the head). Another zipper could be used to close up this cut, but we opted for using Velcro instead.*

usually must refer to a manual, find the next piece and understand how to attach it to their work. In our case, the final assembly is *linear*, i.e., at every instant of the assembly process, the next action is unique and unambiguous, and it requires almost no instructions. In case our method generates self-overlapping strands that must be severed in order to laser-cut them from fabric, sewing the pieces together is simple, since both ends are flat, perfectly matching in length and only requiring flat stitching along a straight line. Furthermore, the zipper replaces the gluing lashes, adhesive tape or other connectors, which can be challenging to work with in free space. To further simplify the alignment of the zipper to the zippable, we propose an optional fastening rig that enables sliding the zipper in and keeping it in place before attaching to the fabric. This could be of particular importance in the context of industrial fabrication, where processes must be streamlined and automated.

We demonstrate our method on various shapes, see e.g. Figs. 5.3, 5.20. We show virtual results and physically fabricated objects, assembled and disassembled simply by zipping up and down.

Figure 5.4: *Our design for a zippable star pillow, made of two differently colored fabrics flatly attached together. Zipping it up generates an interesting interleaving of the two parts. Two of the five Fermat spirals are clearly visible in the top right inset.*

5.2 Method

Given a mesh representing the 3D object, our goal is to generate a single, flat, possibly branching and/or self-overlapping shape, referred to as the *zippable*, which approximates the object when "zipped-up". We can rigorously define zipping-up as an isometric deformation of the flat shape into a 3D shape such that the boundary exactly overlaps with itself. However, we assume that our intention is clear and avoid mathematical rigor at this point. In addition to being "zippable", we wish to enable some creative control by allowing the user to define where the zipper should pass and how it should be oriented or aligned.

Creating a zippable is equivalent to tracing a path – the zipper-curve – on the surface, which forms the boundary of the zippable. Our method is based on the observation that it is trivial to generate a uniform spiral on a cylinder with no caps. We can cut it from one boundary loop (the top) to the other (the bottom) and unfold it to a rectangular shape, where the two boundary curves

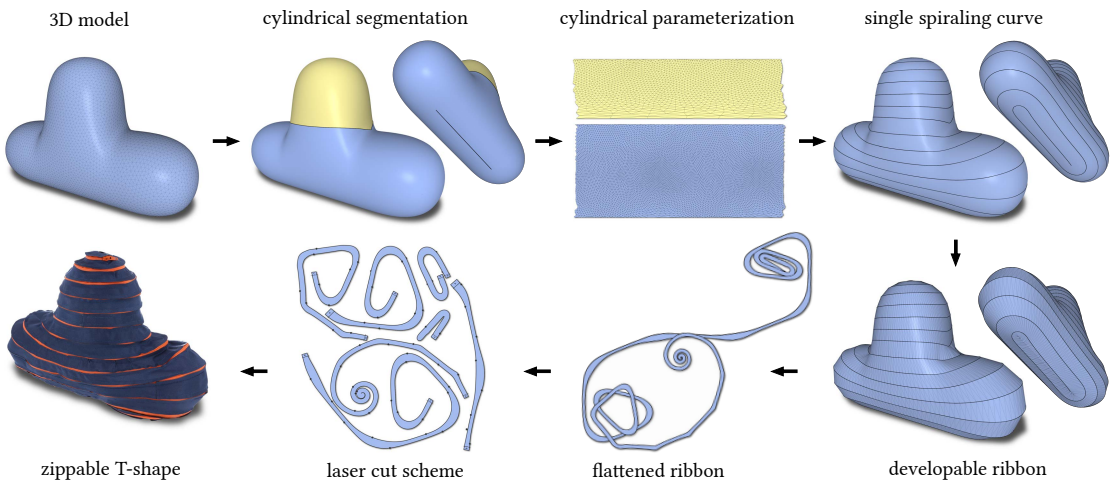


Figure 5.5: *Overview of our pipeline. We begin by segmenting a 3D model to cylinders, followed by a global cylinder parameterization. Using the parameterization, we trace a spiraling curve on the shape. The shape is then cut along the curve and approximated by a developable ribbon. The ribbon is then unfolded to the plane and offset. We proceed by packing the design in order to create a cutting program for a laser cutter. Finally, we cut the design from a piece of fabric and sew a zipper along its boundary. When zipped-up, the ribbon reproduces the original shape.*

become the top and bottom edges, and the cut becomes the two side edges. We then place "copies" of the unfolded cylinder side-by-side, and draw a straight line from the bottom corner of the leftmost edge to the top corner of the rightmost edge, see Fig. 5.6 for an illustration. Overlaying the copies on top of each other creates several disconnected, parallel line segments on the parameterization of the cylinder, and by rolling it back to a cylinder, these segments transform into a perfect connected spiral. Its number of turns depends on the number of copies we made.

The same approach, termed *cylindrical parameterization*, can be applied to general cylinder-like shapes, which we continue informally calling "cylinders". We start in the same manner by cutting the shape from one boundary loop to the other. The cut shape is then mapped to the plane by a distortion minimizing parameterization with *seam constraints*, which force the two sides of the cut to match like puzzle pieces. Minimizing distortion is necessary for the straight line in 2D to be mapped to a smooth and uniform spiral on the surface in 3D. The case of a more complex shape is slightly more involved: we decompose the shape into cylindrical parts and use a global parameterization scheme to smoothly map all the parts to cylinders. We discuss this in more detail in Sec. 5.2.2. Once the mapping is found, we turn to designing spirals on the cylinders. The main challenge is to synchronize the spirals, such that

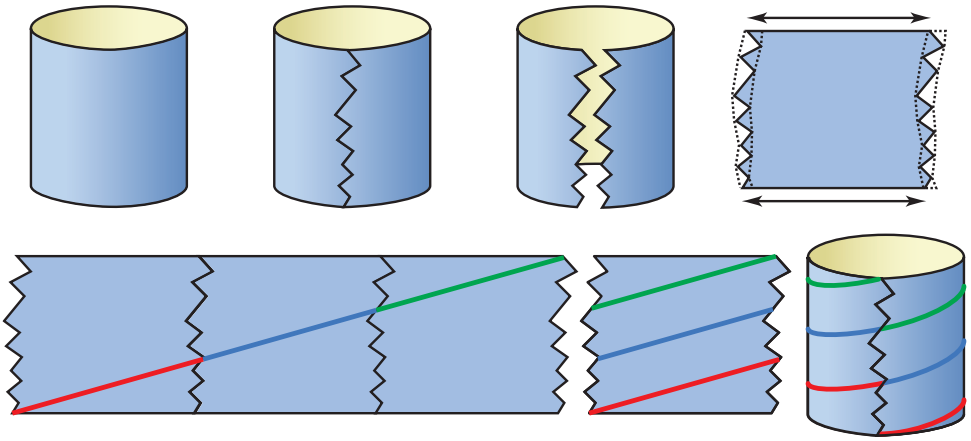


Figure 5.6: *Drawing a spiral on a cylinder can be done by cutting the cylinder from the top boundary to the bottom one and unfolding it to the plane. We then place copies of the flattened cylinder and draw a straight line that passes from the bottom leftmost corner to the top rightmost one. Overlaying the copies and folding back to a cylinder creates a spiral, where the number of windings is equal to the number of flattened copies.*

one spiral ends where another begins, resulting in a single, long zipper-curve on the surface. The final task is to cut the surface along the zipper-curve and remesh it such that the result is developable. It is then trivially isometrically unfolded to generate the final 2D shape of the zippable.

To summarize, the design phase of our method consists of four stages (see Fig. 5.5 for a graphical overview):

1. Decomposition into cylindrical parts.
2. Seamless, global parameterization of the cylinders.
3. Zipper-curve generation.
4. Cutting along the zipper-curve, remeshing and flattening.

5.2.1 Decomposition into cylindrical parts

We partition the input shape S into N *topological* cylinders $\mathcal{C}_i, i = 1, \ldots, N$, i.e., 2-manifolds with two boundary loops. The decomposition plays a substantial role in the final appearance of the spiral, since the resulting curve is aligned to the boundaries of the cylinders. It enables a flexible interface for artistic exploration and is achieved by interactively tracing the boundaries of the segmentation using our software. Since surfaces without boundaries cannot be decomposed into topological cylinders, we also allow the user to cut the shape open and place new boundaries, e.g., small circular holes or

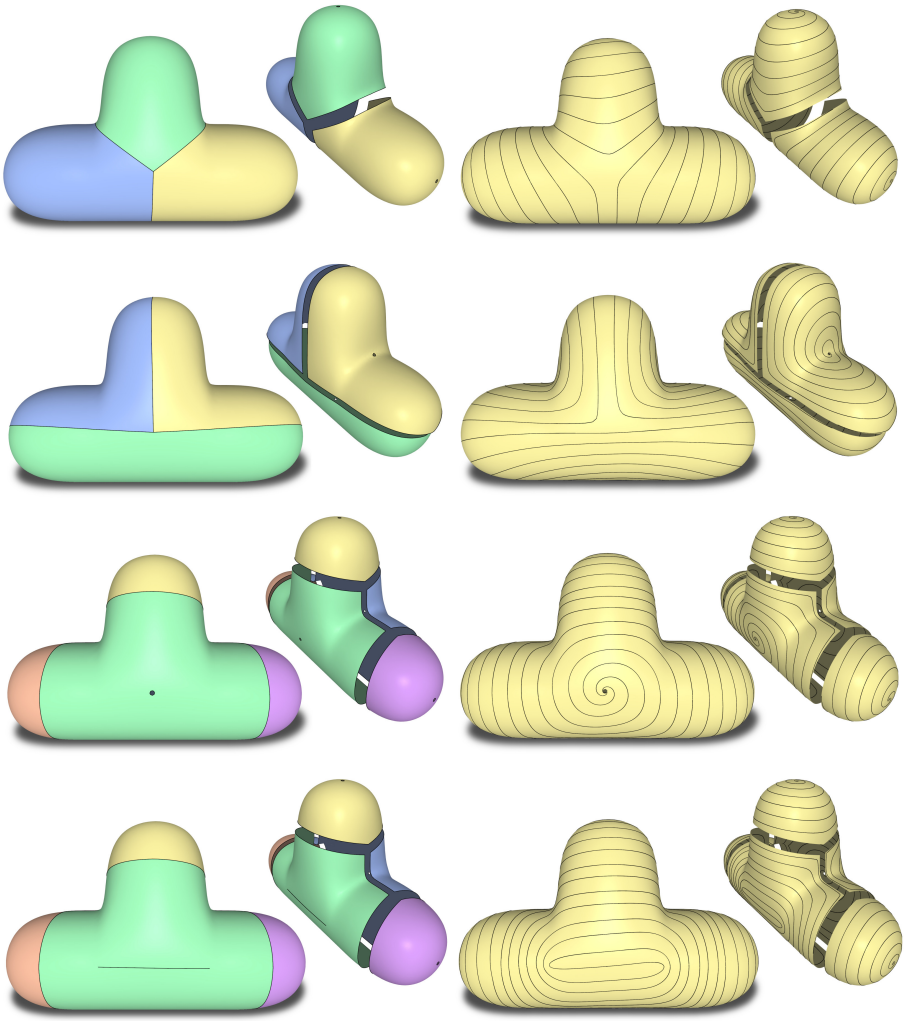


Figure 5.7: *Different cylindrical segmentations of a T shape. Each cylinder has one transition boundary and one open boundary. Note the small holes in the middle of the colored parts. These are the open boundaries for the corresponding cylinders. In the last row we show an example of a straight curve cut, serving as the open boundary of the green cylinder. Compare the resulting spiral to the segmentation in the third row to see the effect of the curve cut.*

curves on the shape that act as a single cylinder boundary (see Fig. 5.7). Our methodology is to start by segmenting the shape into discs, which we feel is more intuitive, and then "puncture" them to obtain topological cylinders. Alternatively, more automatic ways of cylindrical decomposition such as [ZYH+15, LAPS17] could be applied, but we have not explored this option. We distinguish between the *transition* boundary and the *open* boundary of a cylinder. The former being the boundary that separates it from adjacent cylinders, while the later is an actual boundary of the shape (a punctured

hole or curve, as explained above). Further, we define an *interface* to be a shared, continuous edge sequence of two transition boundaries between two cylinders. Therefore, a transition boundary consists of at least one but usually multiple interfaces.

5.2.2 Seamless parameterization

Once the cylindrical decomposition of S into parts $\mathcal{C}_i$ is available, we proceed to compute the parameterization. This step determines how the equally spaced, straight lines in the 2D parameter domain transform into a spiraling zipper-curve on S. To obtain a spiral that is as evenly spaced on the 3D shape as possible, the parameterization must minimize isometric distortion. Additionally, we require the parameterization to be bijective for the mapping from the 2D lines to the 3D curve to be well defined. Fig. 5.8 compares curves generated with the initial (i.e., suboptimal) and optimized parameterization.

We assume that S has been cut along the boundaries of the $\mathcal{C}_i$'s, and each cylinder is cut from one boundary to the other, analogous to the example of one cylinder (see Fig. 5.6 top row). The edges and vertices along the cuts are duplicated, and we keep correspondences between the copies. We generate a seamless bijective parameterization of each $\mathcal{C}_i$, with seamless transitions between adjacent $\mathcal{C}_i$'s. We first explain the case where there is only one cylinder, and the general case of multiple cylinders immediately follows.

Minimizing isometric distortion. An isometric distortion measure quantifies the difference between a given flattening of a shape and a perfect isometry; most formulations define it as a sum of the distortions of individual triangles. In our method we use the recently proposed *symmetric Dirichlet* distortion measure; see [SS15, KGL16, RPPSH17] for details.

We denote the coordinates of a vertex in the parameter domain by $\mathbf{x} = (x, y)$ and stack all the coordinates in a vector $\mathbf{X}$. The distortion of a triangle t is a function of the positions of its vertices in the plane. We denote the symmetric Dirichlet measure of triangle t by $D_t(\mathbf{X})$. Then the optimization problem to solve is

$$\operatorname*{argmin}_{\mathbf{X}} \sum_{\text{triangle } t} A_t \, D_t(\mathbf{X}), \tag{5.1}$$

where A_t is the area of t in the original mesh. In our work, we use a modified Newton's method [SPSH+17] with a feasible starting point to solve this problem. Since we add several constraints in the following, we defer a detailed discussion to the end of this section.

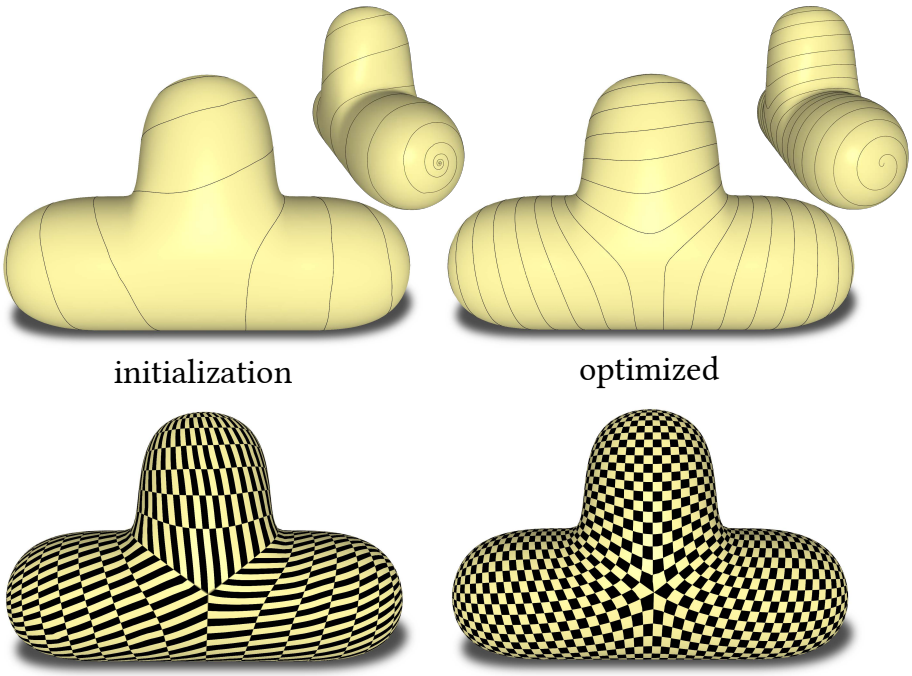


Figure 5.8: *Comparison between the curve obtained before and after minimizing isometric distortion of the parameterization. Note that the non-optimized spirals have a much greater variation in the spacing between the windings. We show a uniform grid texture to illustrate the difference in distortion.*

Seamless cylindrical parameterization. To map a single topological cylinder $\mathcal{C}_i$ to the plane with minimal distortion in a seamless manner, it is cut to form a disk topology and then parameterized while adhering to seam constraints, whose role is to ensure that the parameterization is invariant to the cut [MZ12]. In the cylinder case, the seam constraints call for each edge on one side of the seam to be a translation of its twin edge on the other side of the seam. More precisely, assume the cut contains n consecutive vertices and let $\mathbf{x}_j^{\mathrm{L}}$ and $\mathbf{x}_j^{\mathrm{R}}$, $j = 1, \ldots, n$, be the two copies of each vertex in the parameterization (superscripts L, R stand for Left and Right). Then the cylindrical seamlessness constraints are

$$[\,\mathrm{Cyl}(\mathcal{C}_i)\,] \qquad \mathbf{x}_j^{\mathrm{L}} - \mathbf{x}_{j-1}^{\mathrm{L}} = \mathbf{x}_j^{\mathrm{R}} - \mathbf{x}_{j-1}^{\mathrm{R}}, \quad j = 2, \ldots, n. \tag{5.2}$$

We use the differential form of the seam constraints in order to avoid introducing auxiliary variables. The equivalent positional form is $\mathbf{x}_j^{\mathrm{L}} = \mathbf{x}_j^{\mathrm{R}} + \mathbf{t}$, $j = 1, \ldots, n$, where $\mathbf{t}$ is an unknown offset (the same for all vertices). For conciseness, we refer to the set of constraints in (5.2) as $\mathrm{Cyl}(\mathcal{C}_i)$ for a given $\mathcal{C}_i$, or Cyl in general.

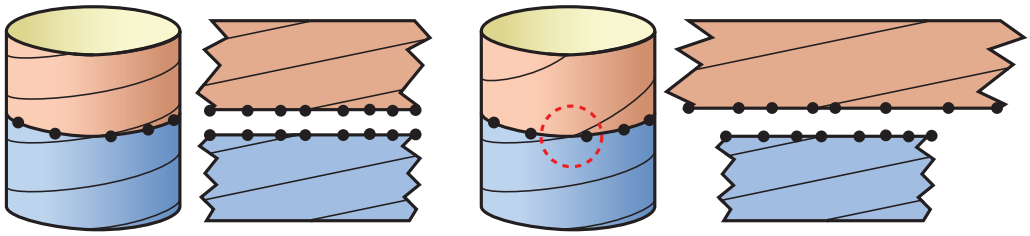
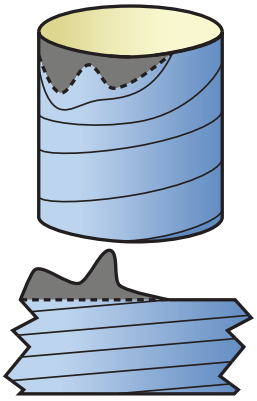


Figure 5.9: *Inter-cylinder constraints ensure that the transitions between cylinders are smooth for curves with the same slopes (left). Note the kink that appears in the curve when these constraints are missing (right).*

Cylinder boundary constraints. In addition to the cylinder seam constraints Cyl, we also require the boundary loops of the cylinders to be mapped to straight lines. This serves two purposes: First, together with the Cyl, it guarantees bijectivity, and second, it allows for a better surface coverage by the spiral. Indeed, when the boundaries are not kept straight and allowed to "spill out" in the 2D domain, the spilled region is not covered by the zipper-curve (see illustration in the inset). Due to Cyl, the straight lines of the boundaries must be parallel, hence, without loss of generality, we can make them parallel to the horizontal x-axis. Let y_k^{Top}, $k = 1, \dots, m^{\text{Top}}$ and y_l^{Bot}, $l = 1, \dots, m^{\text{Bottom}}$ be the y-coordinates of the vertices of the top and bottom boundaries in the parameter domain. We again use the differential form for the straight line constraints, given by



$$[\,\text{Str}(\mathcal{C}_i)\,] \qquad \begin{aligned} y_k^{\text{Top}} - y_{k-1}^{\text{Top}} &= 0, \quad k = 2, \dots, m^{\text{Top}} \\ y_l^{\text{Bot}} - y_{l-1}^{\text{Bot}} &= 0, \quad l = 2, \dots, m^{\text{Bottom}} \end{aligned} \tag{5.3}$$

We denote the constraints of each $\mathcal{C}_i$ in (5.3) by $\text{Str}(\mathcal{C}_i)$, and the entire set of these constraints as Str. The constraints Cyl and Str together already result in a nice spiral on each $\mathcal{C}_i$ separately. However, without dedicated treatment, there could be a visible kink when transitioning between $\mathcal{C}_i$'s (see illustration in Fig. 5.9). Indeed, if copies of the same edge on a transition boundary are parameterized to two edges with different size and direction, then the parameterization does not appear smooth across that edge. We handle this issue in the following.

Inter-cylinder seamlessness. In order for the transition between $\mathcal{C}_i$'s to appear smooth, we apply seam constraints on cuts between each pair of neigh-

boring $\mathcal{C}_i$'s. These constraints enforce a rigid transformation between the two sides of each seam (see Fig. 5.9). Since we have the freedom to define the exact transformation, we choose a rotation by π. Thus, for every two neighboring $\mathcal{C}_P$ and $\mathcal{C}_Q$, we let $\mathbf{x}_r^P$ and $\mathbf{x}_r^Q$, $r = 1, \ldots, s$, be the coordinates of the two copies of each vertex along the seam between $\mathcal{C}_P$ and $\mathcal{C}_Q$. Then the inter-cylinder seam constraints can be written in differential form as

$$\left[\,\mathrm{Int}(\mathcal{C}_P, \mathcal{C}_Q)\,\right] \qquad \mathbf{x}_r^P - \mathbf{x}_{r-1}^Q = \mathbf{x}_{r-1}^P - \mathbf{x}_r^Q, \quad r = 2, \ldots, s. \tag{5.4}$$

We denote the constraints in (5.4) for each pair $\mathcal{C}_P, \mathcal{C}_Q$ by $\mathrm{Int}(\mathcal{C}_P, \mathcal{C}_Q)$.

Global cylindrical parameterization. With all the types of constraints defined, we can formulate the optimization problem for parameterizing the surface:

$$\begin{aligned} \underset{\mathbf{X}}{\text{argmin}} \quad & \sum_{\text{triangle } t} A_t\, D_t(\mathbf{X}) & \\ \text{s.t.} \quad & \mathrm{Cyl}(\mathcal{C}_i), & \forall\, \mathcal{C}_i \\ & \mathrm{Str}(\mathcal{C}_i), & \forall\, \mathcal{C}_i \\ & \mathrm{Int}(\mathcal{C}_P, \mathcal{C}_Q), & \forall\, \mathcal{C}_P, \mathcal{C}_Q \text{ neighbors.} \end{aligned} \tag{5.5}$$

Eliminating degrees of freedom. The constraints in (5.5) are all sparse linear homogeneous equalities, but have some redundancies. For example, it is unnecessary to require all of the top and bottom boundaries to be on straight lines: half of them is sufficient, since the other half then must lie on straight lines due to the Int constraints. Similarly, the y part of (5.5) is redundant due to the Str constraints. We automatically remove all of these redundant constraints using Gaussian elimination.

Initialization. Our optimization is based on Newton's method and requires a feasible starting point with no triangle flips. We use Tutte's embedding with uniform weights, which guarantees bijectivity if the boundary is convex. We can therefore initially map each cylinder to a rectangle in the plane, where the top and bottom boundaries are parallel to the x axis, in order to satisfy Str. We set the height of each rectangle to have the length of the cylinder boundary to get a more isometric initial guess. To satisfy Int we require each edge of a transition boundary to have the same length in its parameterization. We can do the same for the cylinder boundary edges, which then completely determines the boundary. However, we note that we can in fact use Cyl, as they are in the form of an orbifold Tutte's embedding (see [AL15]), instead of

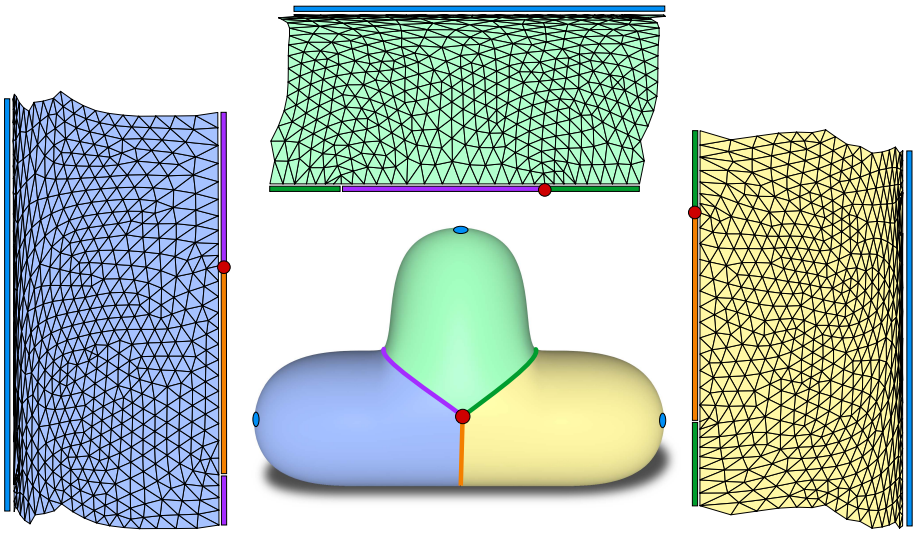


Figure 5.10: *The parameterization of the three topological cylinders of the T shape. See Fig. 5.7 for another perspective of the same segmentation. We mark the corresponding interfaces by matching colors, and the red dot represents one of the two points where all cylinders meet. We remark that this point has no particular significance and is only there as a visual guide. The blue sides of each flattening and the corresponding ellipses represent the open boundaries, while the unmarked sides represent the constrained cylinder seams.*

specifying vertex positions directly, which results in a slightly less distorted initial guess.

Optimization. We use a modified Newton's [SPSH+17] method with linear constraints to solve (5.5). We use the line search method suggested in [SS15], which guarantees that no triangle flips are introduced during optimization. We show an example of the parameterization of the T shape in Fig. 5.10.

5.2.3 Spiraling zipper-curve

With the global seamless parameterization of the cylindrical decomposition available, the next stage in our algorithm is to generate a spiraling curve that represents the zipper-curve. This is done by drawing straight lines in the parameter domain and lifting them back into 3D using the inverse of the parameterization. A spiral on a single $\mathcal{C}_i$ can be created as discussed in the beginning of Sec. 5.2, by drawing a straight line on the cylinder's parameterization. For the case of multiple $\mathcal{C}_i$'s, in order to obtain a single continuous curve, we must make sure that the spirals of the individual $\mathcal{C}_i$'s connect. We make the following simplifying assumptions:

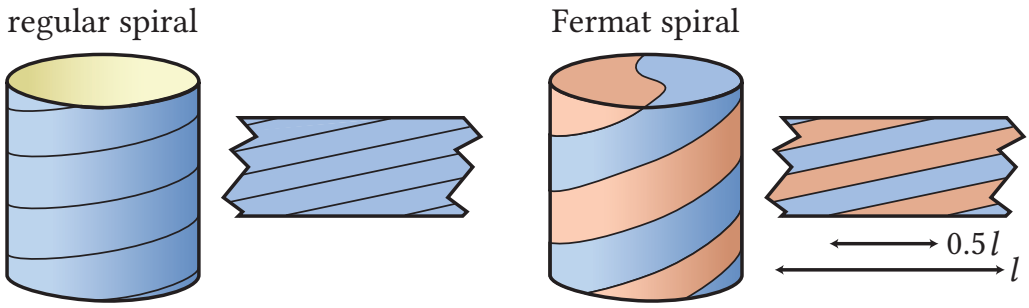


Figure 5.11: *Illustration of a Fermat spiral on a cylinder. The cap of the cylinder on the right represents the open boundary where the center of the Fermat spiral appears.*

1. The curve visits each $\mathcal{C}_i$ exactly once.
2. The curve starts and ends at an open boundary and enters and leaves each cylinder through different interfaces.
3. The curve traverses cylinders via Fermat spirals, except for the first and the last one.

Note that a Fermat spiral requires drawing *two* lines on a cylinder (see Fig. 5.11). The assumptions above mean w.l.o.g. that a curve must start at an open boundary in $\mathcal{C}_1$, then cross the transition boundary through an interface to the adjacent $\mathcal{C}_2$ and touch the open boundary of $\mathcal{C}_2$. Next, leaving the open boundary at another point, passing back through the transition boundary via another interface to $\mathcal{C}_3$. This continues until all cylinders are traversed, and the curve ends at the open boundary of the last $\mathcal{C}_i$. See Fig. 5.12 for an illustration.

There are several choices we let the user make. The first is the traversal order of the $\mathcal{C}_i$'s. To assist the user, we enumerate all valid traversal orders, which are essentially all the Hamiltonian paths on the graph of segments, and let the user choose one by cycling through them. A valid path is guaranteed to exist if the number of segments is smaller than 11 [BJ70]. Excluding pathological cases, the number of possible paths grows dramatically with respect to the number of segments. Many of them can be eliminated by letting the user pick a start and end segment. Another choice is the location on an interface where the curve passes between $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$. Finally, the user can prescribe the number of windings of the spiral on each $\mathcal{C}_i$. See Fig. 5.13 for an example of the different choices.

These choices impact the final appearance of the zipper-curve and should be based mostly on artistic considerations. In general, having even spacing between the windings greatly contributes to the aesthetics of the zipped-up shape. For a regular spiral, even spacing is ensured by the low-distortion parameterization. For Fermat spirals we would like to draw the two lines

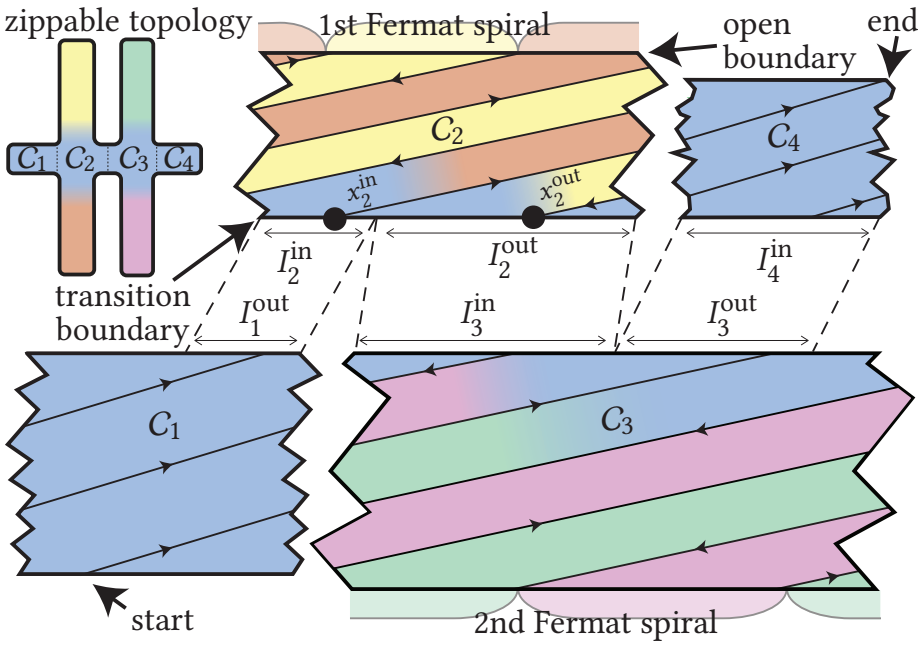


Figure 5.12: *Illustration of a spiraling curve traversing several cylinders. We mark the interfaces of $\mathcal{C}_i$ by I_i^{in} and I_i^{out} for $i = 1, 2, 3, 4$ (see Sec. 5.2.3). The colors help to visualize the zippable's connectivity and relate to the topology of the zippable shown in the top left corner. Every Fermat spiral creates a new branching part shown in red/yellow and green/pink.*

parallel, such that their copies are uniformly spaced. Therefore, the horizontal distance between the lines should be half the width of the boundary (Fig. 5.11). Unfortunately, this is not always possible, since the placement of interfaces between the different $\mathcal{C}_i$'s might not allow it. To illustrate this, consider a cylinder $\mathcal{C}_i$ with $i > 1$, where the zipper-curve comes in from $\mathcal{C}_{i-1}$ through the interface I_i^{in} and leaves to $\mathcal{C}_{i+1}$ through the interface I_i^{out} (see Fig. 5.12, e.g. $i = 2$). The two boundaries (open and transition) of $\mathcal{C}_i$ are parameterized to straight, parallel segments of the same length l_i. For all $\mathcal{C}_i's$ with $1¡i¡N$, we need to pick two points on each segment that will be the end points of the incoming and outgoing lines. For the open boundary, we are free to pick any two points, and the best option is two points with a distance of $0.5\, l_i$. W.l.o.g. we can assume that the interface segments lie on the x-axis. For the transition boundary, we would like to pick two points $x_i^{\text{in}}, x_i^{\text{out}}$ such that

$$\left| x_i^{\text{in}} - x_i^{\text{out}} \right| = 0.5\, l_i. \tag{5.6}$$

However, since $x_i^{\text{in}} \in I_i^{\text{in}}$ and $x_i^{\text{out}} \in I_i^{\text{out}}$, this is clearly not necessarily possible. Even if $I_i^{\text{in}}, I_i^{\text{out}}$ do permit (5.6), we must recall that $x_{i-1}^{\text{out}} = x_i^{\text{in}}$, which thus couples all transition interfaces together. Nevertheless, we can

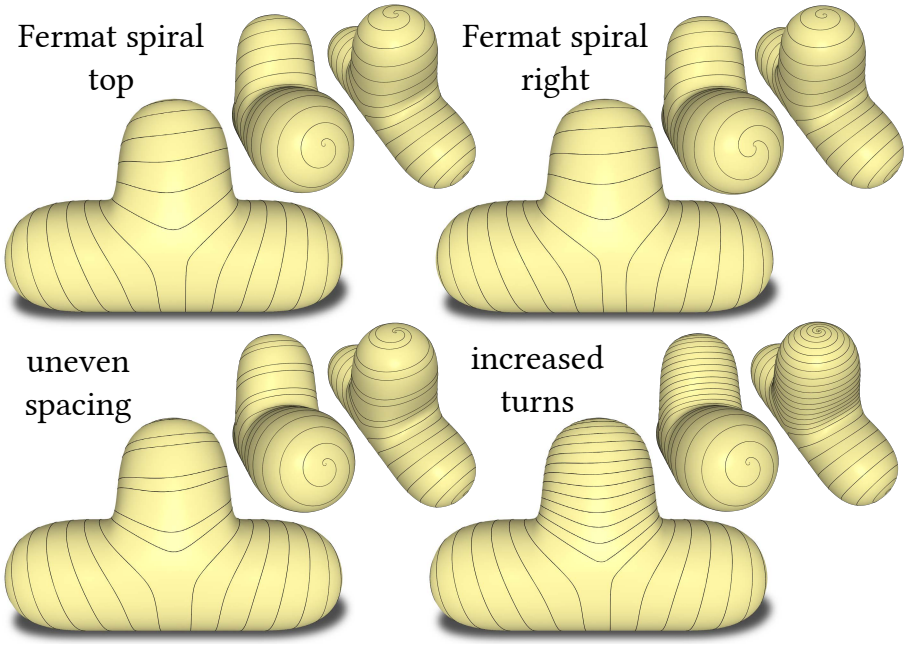


Figure 5.13: *We show several possible zipper-curves on the T shape. The design is up to the user's artistic choices.*

solve an optimization problem to determine the best transition points. We formulate the problem as follows:

$$\begin{aligned} \underset{x_i^{\text{in}}, x_i^{\text{out}}, i=2,\ldots,N-1}{\text{argmin}} \quad & \sum_i \left(\left| x_i^{\text{out}} - x_i^{\text{in}} \right| - 0.5\, l_i \right)^2, \\ \text{s.t.} \quad & x_i^{\text{in}} \in I_i^{\text{in}},\ x_i^{\text{out}} \in I_i^{\text{out}} \quad \text{and} \\ & x_{i-1}^{\text{out}} = x_i^{\text{in}}, \end{aligned} \tag{5.7}$$

where we exclude $\mathcal{C}_1$ and $\mathcal{C}_N$ since they do not contain Fermat spirals. We have the freedom to pick the starting point of each I_i such that $x_i^{\text{out}} > x_i^{\text{in}}$ is always satisfied, allowing to drop the absolute value from the objective in (5.7). There still might be a translational degree of freedom, and we remove it by requiring transition points to be close to the middle of their intervals. In Fig. 5.14 we show the difference between a naive initialization, which tries to keep the zippable width as constant as possible in a greedy way, and the optimized solution.

5.2.4 Cutting, remeshing and flattening

The goal of the final stage of our algorithm is to find a developable surface that approximates the input 3D shape well and has the curve computed in

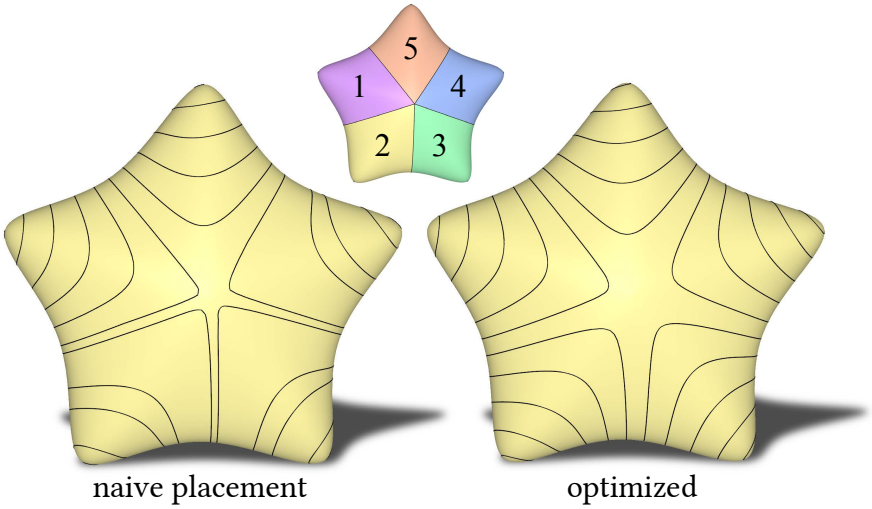


Figure 5.14: *The difference between a naive transition between cylinders, which tries to keep the zippable width as constant as possible with a greedy approach, and the optimized solution. Note that the optimized result appears to be more uniform. The numbers depict the cylinder transition order of the zipper-curve.*

the previous stage as the boundary. This is a challenging task in general (see, e.g., [RSW+07, TBWP16]), but somewhat simpler in the discrete setting. It is well known that a triangle mesh in 3D that has no internal vertices, i.e., all its vertices are boundary vertices, is developable. Fitting a developable triangulation is still a difficult problem, where the challange lies in finding a meshing that appears smooth. Mitani and Suzuki [MS04] proposed to use edge collapse and vertex removal operations until no internal vertices are left, and then to apply edge flip operations in order to improve the smoothness of the triangulation. We have implemented their method, but observed that the greedy edge flipping can introduce triangle fans near sharp curve turns (see Fig. 5.15). We instead propose a simple approach based on the parameterization we already have from previous stages. The idea is to define correspondences between points on adjacent windings of the spiral, which, when connected by straight lines, act as rulings of a developable surface. The simple correspondence we choose is based on the x coordinates of the lines in each $\mathcal{C}_i$'s parameterization (Fig. 5.16). We sample these lines uniformly and connect two samples in adjacent windings when their x coordinates are the same. We use Triangle [She96] to complete the triangulation in the parametrization space in regions where there is no natural correspondence, that is, around the interfaces between cylinders and the open boundaries. Although this simple approach is not always optimal (see Fig. 5.24), we found it to deliver sufficient results for all cases. Once the zippable is triangulated, we can unfold it to the plane (see Fig. 5.5). We cut it into few smaller pieces in

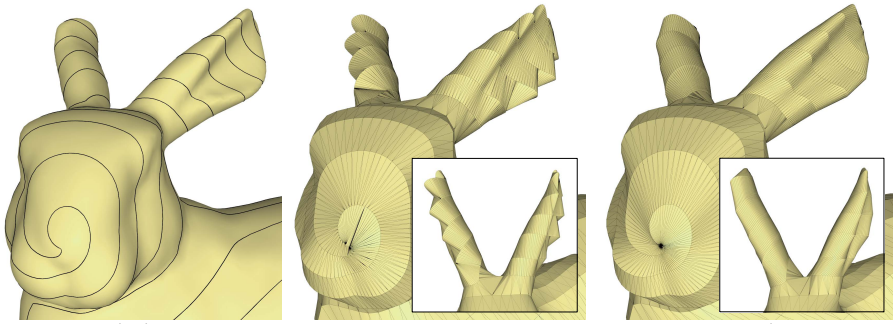


Figure 5.15: *A comparison of our simple ribbon meshing approach to the method in [MS04]: The greedy edge flipping step can generate non-optimal triangle fans (middle column), whereas ours results in a smoother and better approximation of the original surface. But generally, it is not guaranteed to create an optimal meshing (see Fig. 5.24).*

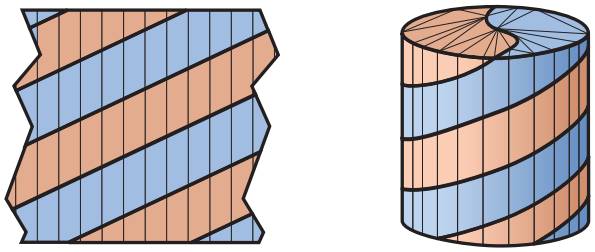


Figure 5.16: *Illustration of remeshing to a developable zippable. In the parameter domain, points on adjacent lines with the same x coordinate are connected by an edge.*

order to utilize the laser cutter bed better and resolve overlaps. The minimal number of cuts is related to the number of segments (parts from different segments might intersect) and the number of turns in each segment (different turns in the same part might intersect).

5.3 Fabrication

Zipper tape. The previous section describes how to design and compute a zipper-curve on the surface, but ignores the physical properties of the zipper itself. This is sufficient if one wishes to make papercraft, as in Fig. 5.21, since the zipper-curve has a negligible width. Accommodating a real zipper requires additional modeling. In general, zippers are made from two fabric tapes with a row of teeth on each (Fig. 5.17), which interlock or split when operating the slider. The common way to attach zippers to fabric is to sew the two tapes onto the matching edges of two parts. The teeth should slightly protrude to create a wide enough gap between the two pieces of fabric, so

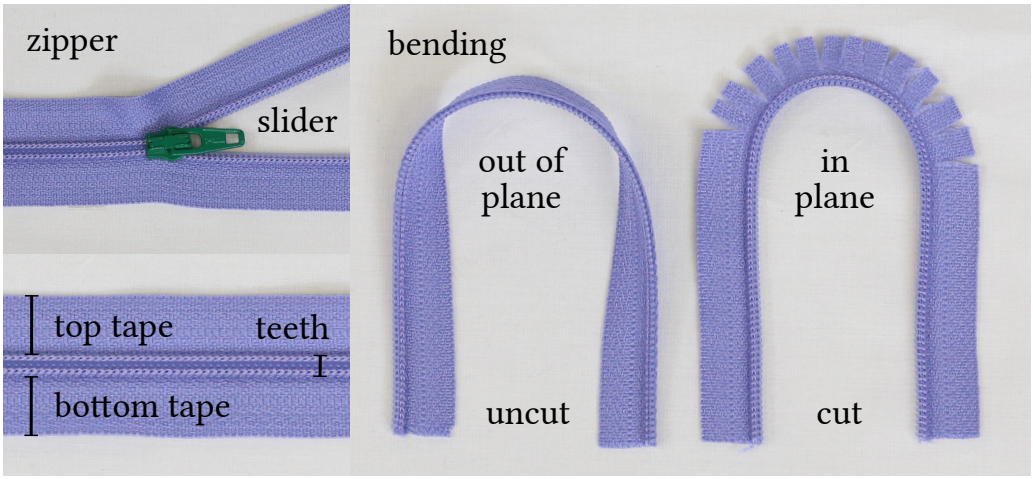


Figure 5.17: *A dissection of a zipper. Note that the zipper resists bending in the plane, unless cut every few centimeters to allow the zipper tape to stretch.*

that the slider can move freely without getting stuck. This means that we must slightly offset the borders of the designed zippable in order to create that gap. We discuss offsetting in later subsections.

Boundary alignment. To ensure that corresponding borders are aligned with each other through the zipper, we place markers (e.g., small circles) at regular intervals along the boundary of the zippable and on the zipper tape. We use 3 cm intervals; the markers can be etched by the laser cutter. When attaching the zipper, care must be taken to align the markers on the zippable with the ones on the zipper tape. After the zipper is attached, we align the starting points of the two sides of the zipper and put the slider in to obtain the final result.

Overlapping zippables. If the 2D layout of the zippable contains any overlapping regions or is too big for the available laser cutter or fabric, we separate it by cutting into intersection-free parts. These parts are then marked for etching with corresponding letters, arranged efficiently in the plane, laser-cut and sewn back together.

5.3.1 Attaching the zipper.

In this section we describe the fabrication process in more detail. We propose two different methods that differ by the offsetting procedure and the zipper attachment technique, with each method having its own benefits and

drawbacks. The first approach is based on sewing the zipper to the fabric, and the second approach is based on gluing and requires the zipper and fabric to be mounted on a bespoke fastening rig. The sewing approach can be used for larger target sizes, where the width of the zipper can be largely neglected. It requires more expertise and manual work, but allows for more flexibility in fabrication. The second technique is particularly handy when the zipper width is not negligible, or for fabrication in an assembly line, since it comprises several sequential steps.

Attaching the zipper by sewing To the best of our knowledge, there are currently no devices for automated sewing of zippers along a curved path, so this must be done manually using a sewing machine. Sewing on a curve is somewhat complicated since zippers are usually designed to have zero geodesic curvature. Their straight tapes resist bending in the plane (although bending out of plane is possible, as we discuss in the following section). To afford this bending, the tape must be cut every few centimeters, as shown in Fig. 5.17.

2D-Offsetting. Since the zipper teeth have to protrude by a certain distance w, we must compensate for that by offsetting the boundary curve of the zippable by the same amount. This can be easily done in the plane using a standard curve offset in the normal direction by distance w. The only caveat is that this changes the boundary length, such that markers on the offset curve are no longer 3 cm apart. As a consequence, while sewing, the tailor needs to ensure that the markers on the zipper and the fabric still line up by slightly squeezing the fabric or the zipper tape, or cutting them, as shown in Fig. 5.17. For larger objects, the resulting inaccuracies are fairly small, as can be seen in the results in Figures 5.4, 5.20, 5.3, 5.5, which were all created using this approach. The fabricated shapes remain faithful to the design.

Attaching the zipper by welding or gluing In modern clothing industry, zippers are often attached by welding or gluing to the underlying fabric, especially in outdoor, waterproof garments and equipment. This can be beneficial for the fabrication on an assembly line, since the different working steps can be split better than for sewing, which is still one of the most manual labor-intensive industries today. Gluing may also be a preferable technique for makers with no sewing experience or the necessary machines. It requires the fabric and the zipper to be first placed and secured in correct alignment before applying the necessary pressure to fixate it. Therefore, one can no

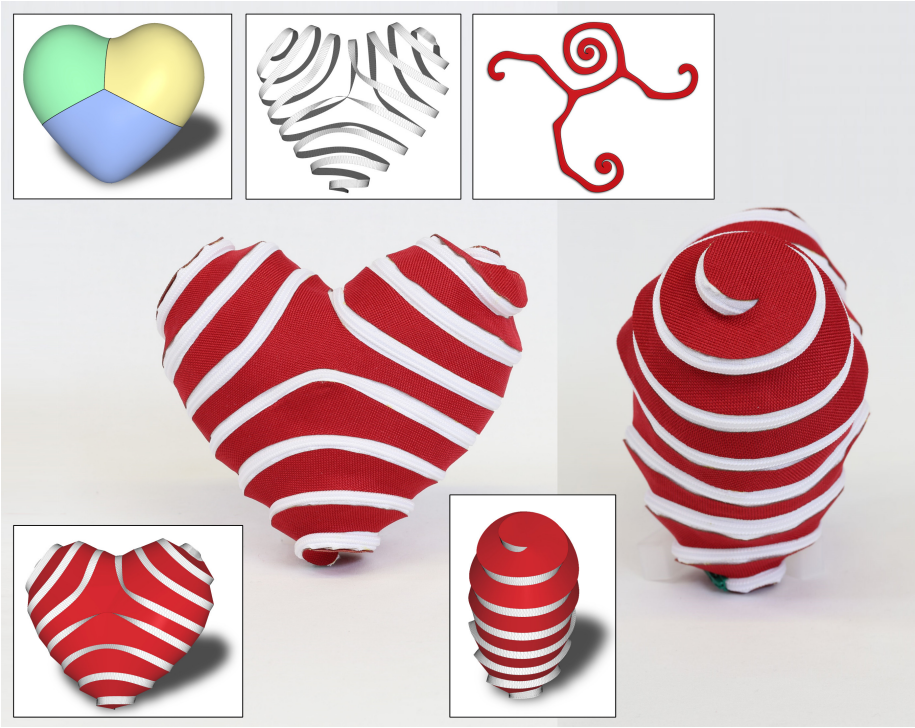


Figure 5.18: *The heart is our smallest physical result with a height of only 15cm. It was fabricated using our fastening rig in only 1 hour. The virtual model (insets in the bottom) is faithfully reproduced. The top insets show the segmentation, the visualization of the zipper tape and flattened zippable, from left to right.*

longer assume that local squeezing is freely permitted, which calls for a different offsetting strategy.

Length preserving offsets. Forbidding the zipper (and the fabric) from stretching means that it can only undergo isometric deformations, which prohibits the planar curve offsetting we proposed for the sewing method. However, in practice and as shown in Fig. 5.17, the zipper has the ability to bend out of plane. Therefore, we consider the offsetting in 3D (see Fig. 5.18). In reality, the zipper tape can also easily twist and shear a bit, so instead of precisely isometric deformation, we make a simplification and assume that the two sides of the zipper, or equivalently the two sides of the offset boundary curves of the zippable, only need to have the *same length* to be a valid zipper configuration. Given the zipper-curve on the surface in 3D, we seek two curves that are offset by the same amount in *opposite directions*. We propose to use the *binormal* of the zipper-curve as the offset direction. We prove in Appendix C that in the continuous setting, the offsetting in the positive and negative direction is guaranteed to keep the lengths of the

two offset curves equal, and we bound the length difference between the zipper-curve and the offset curves. In practice, this small difference can be compensated by a slight buckling of the zipper tape. In the discrete setting, we estimate the binormal by the cross product of two adjacent segments of the zipper-curve. Note that the binormal direction for a straight line is not well defined, and so in regions where the polyline is almost straight, we might get numerically noisy results. To avoid this, we consider a bigger window of adjacent segments to find a stable estimation of the binormal at these locations, at the cost of a small deviation in length. To validate our results, we compute the relative change in length that occurs due to the approximation. For the heart shape in Fig. 5.18, we use 2.25 meters of zipper, and our error of 0.2 mm is negligible. Two results with this type of offsetting are shown in Figures 5.18, 5.2.

The fastening rig. Central to the gluing approach is a bespoke fastening rig we developed. The idea is to secure the zipper and the zippable in place before applying contact glue and pressing them together. In order to glue the zipper tape completely flat onto the zippable, one needs to cut it at regular intervals. The important difference to sewing is that this does not result in any buckling or deformation of the zippable and is completely hidden behind it. The rig is constructed from 3 layers of hard sheets (e.g., acrylic glass or plywood) as shown in Fig. 5.19. The bottom one serves as a baseplate to stabilize the others. We cut tracks into the middle sheet for sliding in the zipper tapes. The tracks have the shape of the boundary of the flattened zippable and a width of about 4 mm. The teeth track is curved, but it only bends the zipper tape out of plane. The cut in the top layer is slightly offsetted, such that the zipper tape can pass through the gap but the teeth are held down. We also add an opening for the tracks to simplify the sliding in of the zipper in parts. The layers are connected and secured with screws. Even though the rig does a good job in aligning the zipper tape to the zippable, we still add markers in 5 cm intervals on the zipper tape and the rig as a detailed alignment guidance. If the zippable has overlaps, the tracks for the shape can also be split into multiple parts as before, which then only requires to slide in and glue the zipper to the zippable part by part (the result in Fig. 5.2 was produced in this way). We found that this adds no significant time to the fabrication process. See Fig. 5.19) for more detail.

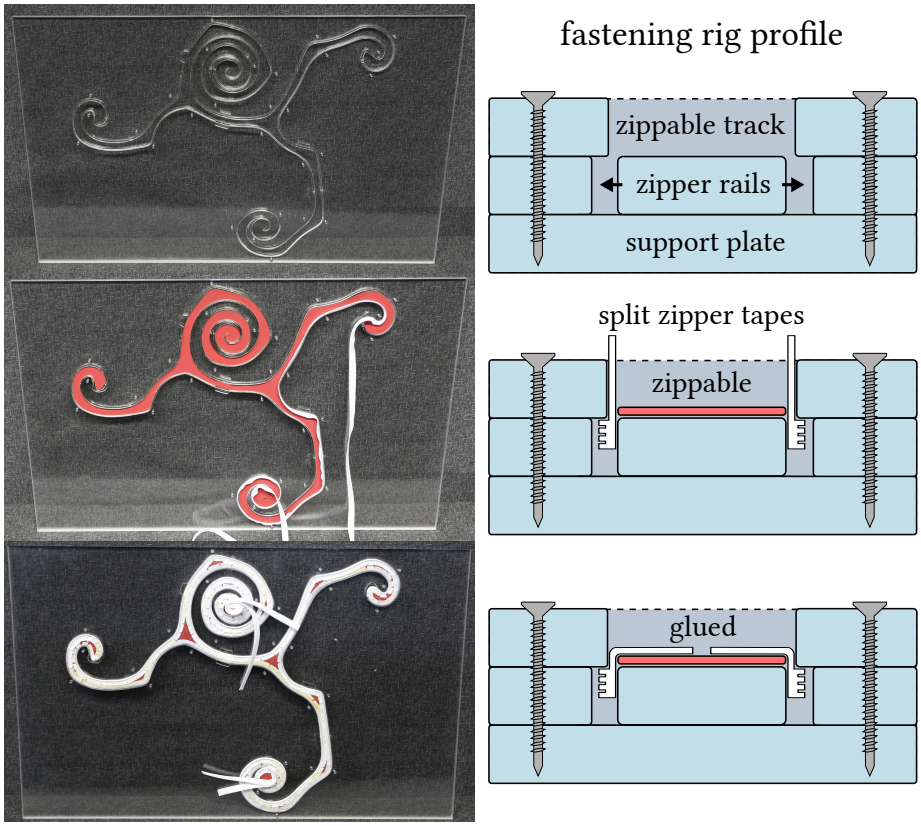


Figure 5.19: *An overview of the fabrication with the fastening rig. From top to bottom: the empty assembled rig; rig with inserted zippable and partly slid in zipper tape; the completed zippable. Sliding in the zipper tapes is easy and fast. The tracks in the fastening rig almost automatically take care of the correct alignment of the tape to the zippable.*

5.4 Results

We implemented all parts of our algorithm in C++, except the parameterization, which was implemented in MATLAB. The modified Newton's method converges within at most 15 iterations and takes about 20 seconds for a mesh of 30k triangles on our Xeon CPU E5-2650 v2, 64 GB RAM machine.

Fabrication process. We fabricated seven of our designs in fabric, see Figures 5.1, 5.2, 5.3, 5.4, 5.5, 5.20, 5.18. Fig. 5.5 shows the steps of our method with our simplest model, the T shape. All fabricated results have a zipper length of about 10 meters and a maximum dimension of about 50 cm, except the heart (Fig. 5.18) with 3.5 m zipper and 15 cm height, and the star (Fig. 5.2) with 2.25 m zipper and 27 cm height. These last two are considerably smaller and were fabricated with our gluing technique. The sewn results were made

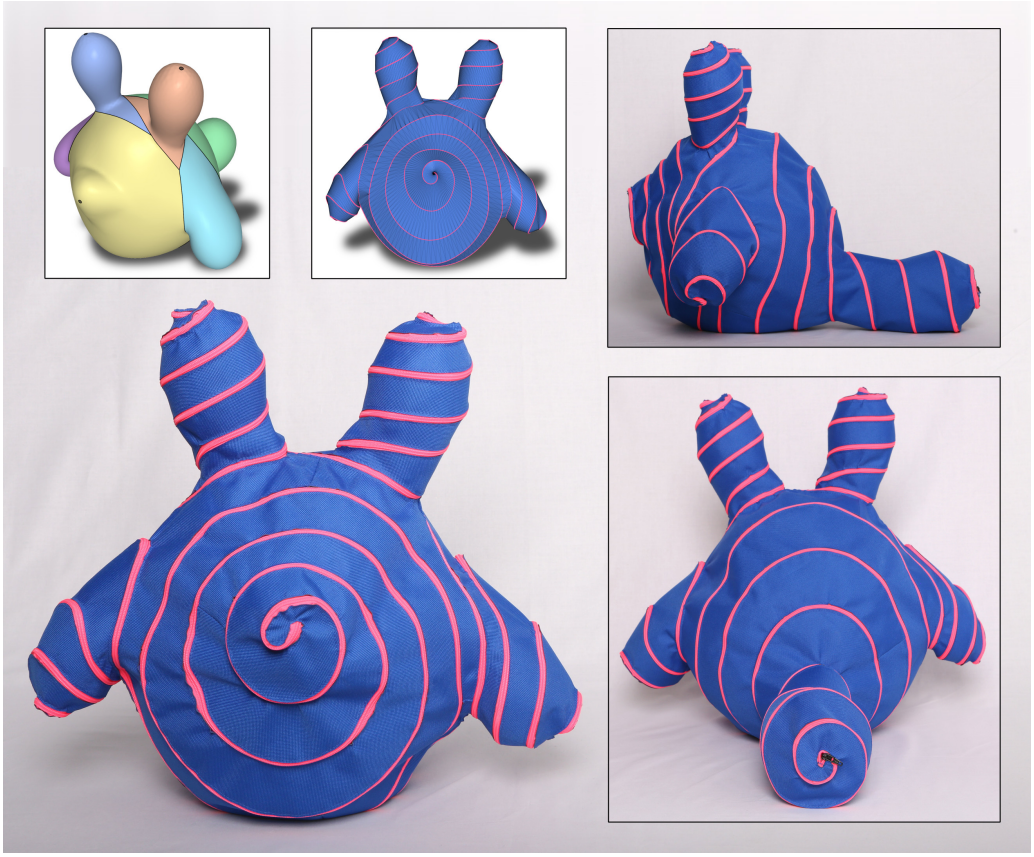


Figure 5.20: *A zippable model of an anime character. The zipper starts at the tail and spirals around all extruding parts until it ends at the tip of the nose.*

by professional tailors and took 5 to 6 hours for each model. The time needed mainly depends on the length of zipper-curve. Even though the tailors had no experience with this special kind of fabrication, there were no problems in attaching the zippers to the cut fabric pieces thanks to the linearity of the assembly method. In comparison, the fabrication of the results made with the bespoke fastening rig approach took us 1 hour for the heart and 1.5 hours for the star shape. Cutting and constructing the fastening rig amounts to about 30 min overhead, and the rig can serve the making of many copies of the same shape.

Design process. We designed all the presented examples ourselves. Simple designs like the heart (Fig. 5.18) or the star (Fig. 5.2) can be made within only two to three minutes. More complicated models like the octopus (Fig. 5.23) can take up to 15 minutes. Most time is spent iterating and refining the cylindrical segmentation in order to optimize the zipper-curve shape, since the global parameterization has to be recomputed each time. In the future, we

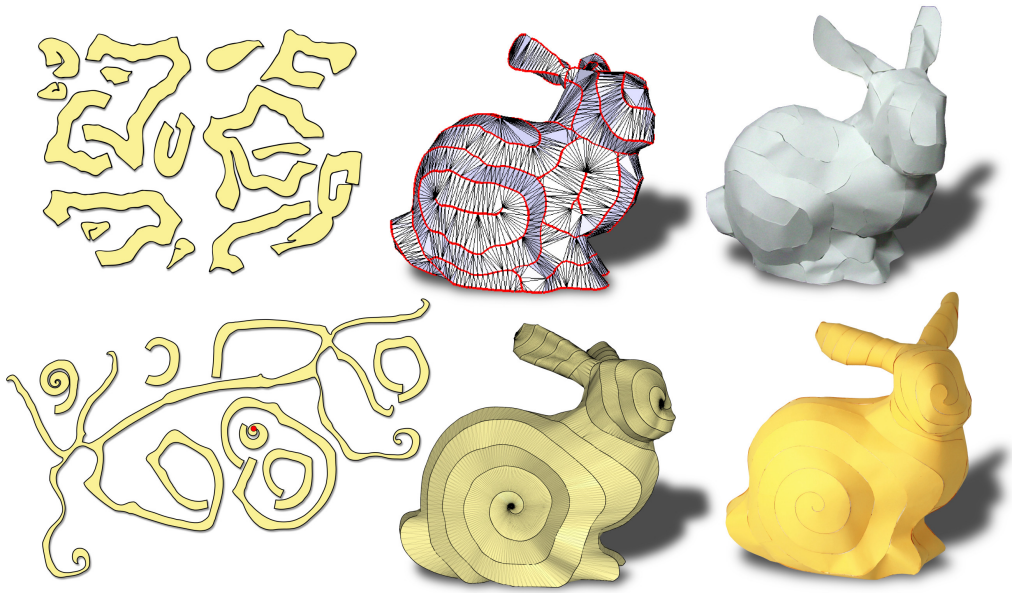


Figure 5.21: *Our laser cut plans of the bunny with 7 pieces (bottom row) for a single developable piece that can be assembled by linearly gluing its border, starting at the designated red point, compared to [MS04] (taken from their paper) with 15 pieces which require more detailed instructions to be glued together (top row). Note that even though our result is somewhat finer in terms of the width of the parts, it is fabricated from fewer pieces.*

would like to explore splines instead of straight lines in the parameterization to enable a more flexible zipper-curve design, possibly aligning it to geometric features or user prescribed directions. While the transition point optimization (Sec. 5.2.3) finds the best possible solution for a selected traversal order, it is not guaranteed to always create a nice, uniformly spaced zippable. This is due to the limited degrees of freedom of the corresponding traversal interfaces. In this case, the user can iterate through the other traversal orders to pick a better choice. In the future, it would be interesting to incorporate an automatic search to find the best possible traversal order in terms of uniformity of the zippable. Another useful feature would be a way to bound the maximal curvature of the zipper-curve to make it smoother.

Papercraft comparison. Our method can be used to create papercraft models. In general, it produces fewer initial pieces than previously published methods, and the assembly by gluing is straightforward and does not require elaborate instructions. See Fig. 5.21 for a comparison of our bunny result fabricated from paper with the method of [MS04]. Note that even though our

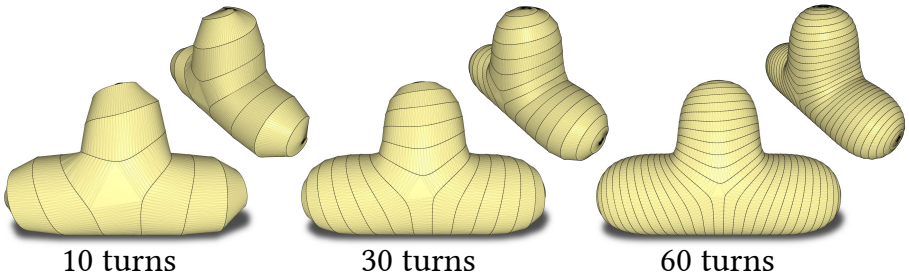


Figure 5.22: *Running our method on the T shape with different spiral densities.*

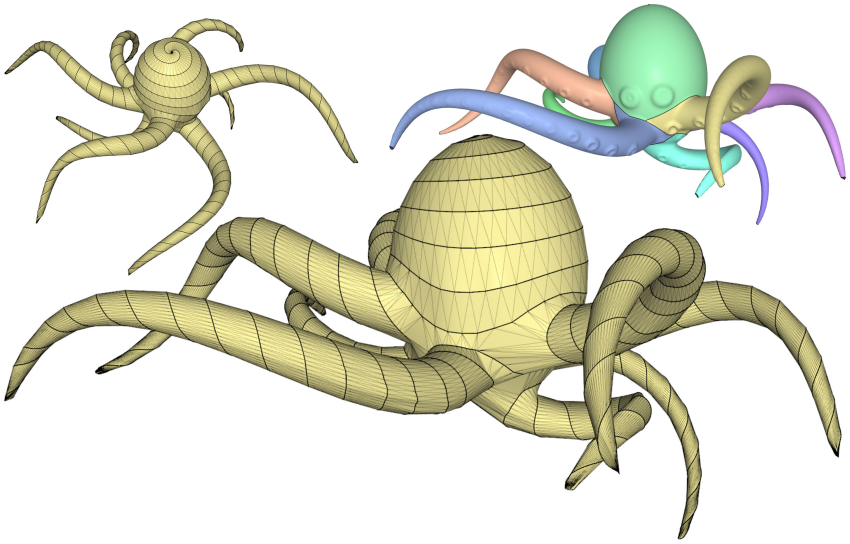


Figure 5.23: *A virtual result of an octopus model segmented into nine cylindrical parts. The zippable has a nice uniform spacing but some of the geometric details, like the eyes, are lost due to the limited resolution.*

result is somewhat finer in terms of the width of the parts, it is fabricated from fewer pieces.

Approximation capabilities. Naturally, the thinner the designed zippable, the better its approximation power. However, this also results in a longer zippable, prolonging the fabrication. The decision regarding the sizing is left to the user. See Fig. 5.22 for different widths and approximations of the T shape.

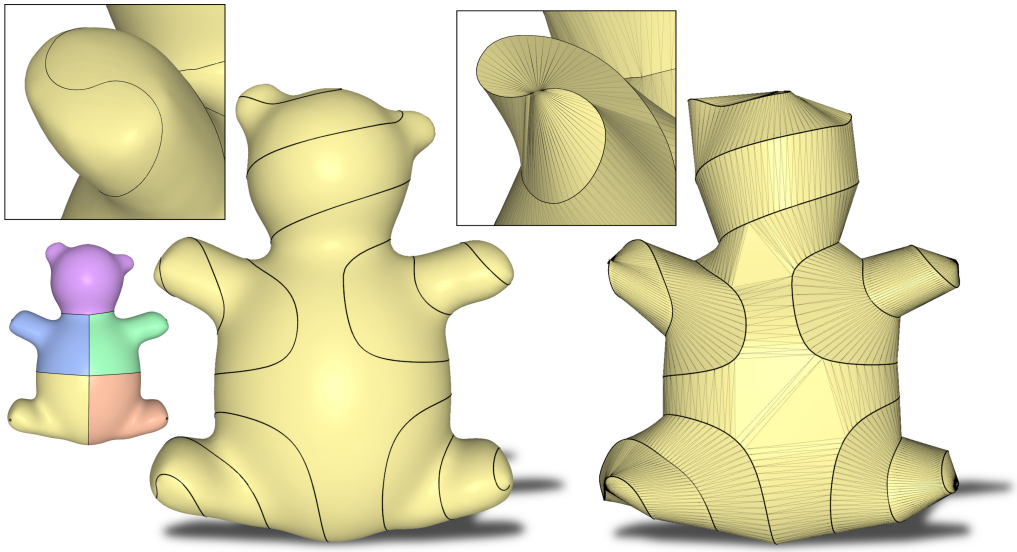


Figure 5.24: *Our naive meshing algorithm is not guaranteed to produce an optimal approximation of the original surface, especially for a sparse coverage of the target model with the zipper-curve.*

5.5 Concluding remarks

We presented a method for shape representation by a single developable surface that can be fabricated from flat fabric. We show several examples to demonstrate the power and generality of our approach. Currently, we do not attempt to align the zipper-curve to the input shape's features. This means that regions with sharp corners may not be well represented by the zippable, unless the user manually specifies it. We plan to tackle this issue in the future by using feature detection and incorporating it into the zipper-curve design stage. Additionally, we are interested in targeting more global objectives, such as symmetry. Furthermore, we would like to combine the zipper-curve design with the final stage of remeshing to a developable. Currently these steps are strictly decoupled, and we expect to get better approximation quality by optimizing both parts simultaneously. Another interesting direction would be to integrate the optimization into the segmentation stage, such that the design of the final zipper-curve can be done more interactively.

Fabrication with textiles, especially woven fabrics, is a very prolific but currently notoriously human labor-intensive industry, in dire need of digitalization and automation. We plan to explore further automation and acceleration of our fabrication process and expand to other types of fabrication methods and applications. One application we are particularly interested in is *pipe cladding*, which is part of the process of insulating heated pipes with

metal sheets, and is a labour intensive task. The process is similar to our zipping, but usually performed manually by an expert; our method could be potentially used to greatly simplify and speed up this task.

CHAPTER 6

Conclusion

6.1 Discussion

In this thesis we developed different novel computational methods to advance traditional crafting and manufacturing techniques. Our work is embedded in the field of computational design and fabrication, including computer science, engineering, materials science, architecture, human-computer interaction, robotics and more. The proposed techniques empower makers as well as professionals to fabricate 3D objects from digital models with a new level of complexity and quality. The chosen approaches either improve upon existing techniques (Chapters 3, 4) or explore and apply new concepts (Chapter 5) to push the boundaries of what is possible in computational fabrication. We empirically verified and showcased the validity and quality of these approaches through the fabrication of many example results.

In Chapter 3, we propose an approach for the creation of reliefs from digital models that is more general and allows for finer control than what previous work could achieve. Our Laplacian-based deformation method provides a unified solution to create all types of reliefs and even extends it to concepts from art, related to anamorphic illusions. Since the input only consists of a 3D model, the back-plane geometry and the constraint volume, no artistic skill or experience is necessary to create the reliefs. Naturally, there is usually a trade-off between full automation and interactive control; in our case, simplifying the design process for arbitrary 3D reliefs required an expensive optimization,

which adds to the design time. In practice, the choice between automation and interactivity typically depends on the application.

In Chapter 4 our contribution is twofold. Fabricating colored 3D objects by thermoforming requires an easy-to-use hardware setup as well as a suitable software solution. We developed an approach that considers both, with the aim of making this process accessible to the maker community. The hardware pipeline setup is built from off-the-shelf components and the computational method requires no further input than the model geometry after a one-time calibration. We used a computer graphics-inspired elastoplastic material model that can be simulated much more efficiently then existing industrial solutions. In general, physically-based simulation methods from the field of computer graphics often pose an interesting alternative to classical simulation problems if efficiency is prioritized over accuracy. We found that our approach garnered attention from individual makers, but also from the thermoforming industry, indicating that the provision of computational algorithms for this kind of fabrication can benefit both.

The work on *zippables* in Chapter 5 was inspired by the simple concept of *zipit* bags [zip17]. While the problem of creating a single, regularly spaced, smooth zipper-curve on a model's surface is easy to understand, we found that coming up with a reasonable solution is far from trivial. As in many other cases, the key lies in the right choice of coordinate representation. Using tools and insights from mesh parameterization, we were able to cast it as a novel global cylindrical parameterization problem. But reality can be unforgiving and carefully devised digital designs, once fabricated, quickly reveal shortcomings in the initial method. In our case, we found that the zipper tape's dimension needed to be taken into account, since it plays an important role in the final assembly process.

In fact, during the course of this thesis we learned that computational methods and algorithms for fabrication should always be informed by reality. Every model makes approximations and assumptions, which ultimately need to be validated and tested on real-life examples.

6.2 Future directions

The field of computational design and fabrication has been very active in recent years. New design algorithms and innovative compact digital fabrication devices enabled the fast growing maker community to explore new ways of manufacturing. Our work contributes to this development and helps to inspire novel ideas and solutions. Every new technique not only has an

immediate impact on the intended application, but also often triggers new developments in other areas.

The core of digital relief creation, for example, is very related to the problem of depth range compression in 3D movies. Making it more interactive could be another interesting direction to increase its applicability. Furthermore, the proposed thermoforming method could be used to create better rear-projection screens of non-planar shape. The applications are manifold and range from artistic lamp design to the production of animated faces for theme park characters. The parameterization approach taken in the design and computation of *zippables* might turn out to be useful for research in quad-meshing as well. In the industry, cladding could be a possible alternative application, where insulation is wrapped around pipes and pre-manufactured pieces could simplify the manual assembly process. A quick look through the related work section reveals that there still exists a number of unexplored manufacturing techniques which could benefit from new computational design approaches, e.g. weaving with yarns, hot-wire foam cutting, pottery from turntables or blow-forming, just to name a few.

In general, it is difficult to predict the final impact of individual works in research. In particular, a fresh PhD student has a hard time to foresee the full potential of a specific research question. Should it be driven by the search of new applications for cutting edge techniques like, e.g. deep reinforcement learning, or is it better to pursue a grand vision while being completely open about the actual method? Personally, I think it is always wise to work on something that is close to the heart, no matter which of the former approaches is taken. Good solutions and ideas cannot be forced but are the rewards for passionate hard work, when they finally emerge in unexpected ways and moments.

APPENDIX A

Quadratic programming problem conversion

We use MOSEK [AA00] to efficiently solve sparse, quadratic programming problems. Its documentation strongly recommends converting convex quadratic energy minimization with linear inequality constraints, like Eq. (3.14), into linear energy minimization with conic constraints. We found this to be especially advantageous for our problem, which is of the form:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{F}\mathbf{x}\|^2 + \mathbf{f}^\mathsf{T}\mathbf{x} + \text{const} \\ \text{subject to} \quad & \mathbf{C}_I\,\mathbf{x} \leq \mathbf{d}, \;\; \mathbf{C}_E\,\mathbf{x} = \mathbf{b} \end{aligned} \tag{A.1}$$

with

$$\begin{aligned} \mathbf{x} &= [\lambda \;\; \mu]^\mathsf{T} \\ \mathbf{f} &= \mathbf{0} \\ \mathbf{F} &= \begin{bmatrix} \mathbf{D}_A\,\mathbf{D}_w(\tilde{\mathbf{L}}^0\mathbf{D}_{\hat{\mathbf{V}}} - \mathbf{D}_{\mathbf{L}_\theta})\mathbf{S} & \mathbf{0} \\ \mathbf{0} & \sqrt{\alpha}\cdot\mathbf{I} \end{bmatrix}. \end{aligned}$$

The matrix $\mathbf{I}$ is an identity matrix of size of the length of μ. To convert this to a conic problem, we first introduce a vector of auxiliary variables $\mathbf{t}$ and rewrite Eq. (A.1) as:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{t}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{t}\|^2 + \mathbf{f}^\mathsf{T}\mathbf{x} + \text{const} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{d}, \;\; \mathbf{C}_E\,\mathbf{x} = \mathbf{b}, \\ & \mathbf{F}\,\mathbf{x} - \mathbf{t} = \mathbf{0}. \end{aligned}$$

Using the scalar variables v and c we convert into conic form:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{t},v,c}{\text{minimize}} \quad & v + \mathbf{f}^\mathsf{T}\mathbf{x} + \text{const} \\ \text{subject to} \quad & \mathbf{C}_I\,\mathbf{x} < \mathbf{d}, \;\; \mathbf{C}_E\,\mathbf{x} = \mathbf{b}, \\ & \mathbf{F}\,\mathbf{x} - \mathbf{t} = \mathbf{0}, \\ & cv \geq \sum_i t_i^2, \;\; c = 2, \;\; v \geq 0, \end{aligned} \tag{A.2}$$

where the inequality constraint on v forces its value to be inside the cone described by the coordinates of $\mathbf{t}$. Putting all variables in a column vector, we can write this in matrix form, as we supply it to the solver:

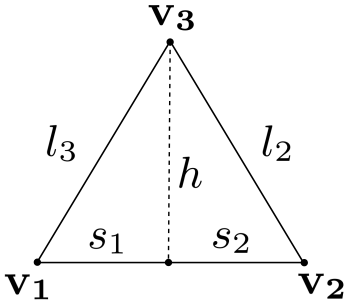
$$\begin{aligned} \underset{[\mathbf{x}^\mathsf{T} \;\; \mathbf{t}^\mathsf{T} \;\; v \;\; c]}{\text{minimize}} \quad & \begin{bmatrix} \mathbf{f}^\mathsf{T} & \mathbf{0}^\mathsf{T} & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{t} \\ v \\ c \end{bmatrix} + \text{const} \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{F} & -\mathbf{I} & 0 & 0 \\ \mathbf{C}_I & \mathbf{0}^\mathsf{T} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{t} \\ v \\ c \end{bmatrix} \geq \begin{bmatrix} 0 \\ -\infty \end{bmatrix} \\ & \begin{bmatrix} \mathbf{F} & -\mathbf{I} & 0 & 0 \\ \mathbf{C}_I & \mathbf{0}^\mathsf{T} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{t} \\ v \\ c \end{bmatrix} \leq \begin{bmatrix} 0 \\ \mathbf{d} \end{bmatrix} \\ & \begin{bmatrix} \mathbf{x} \\ \mathbf{t} \\ v \\ c \end{bmatrix} \leq \begin{bmatrix} \mathbf{b} \\ \infty \\ \infty \\ 2 \end{bmatrix}, \quad \begin{bmatrix} \mathbf{x} \\ \mathbf{t} \\ v \\ c \end{bmatrix} \geq \begin{bmatrix} \mathbf{b} \\ -\infty \\ 0 \\ 2 \end{bmatrix} \\ & cv \geq \sum_i t_i^2. \end{aligned}$$

Note that every equality constraint was replaced by two inequality constraints. Because in our case $\mathbf{C}_E$ is a diagonal matrix, we can represent these equality constraints with an upper and lower bound on $\boldsymbol{\lambda}$, which can be handled more efficiently in the optimization.

APPENDIX B

Vertex positions from edge lengths

Given the three edge lengths of an arbitrary triangle, we reconstruct its vertex positions up to a rigid transformation. W.l.o.g., we place the vertices $\mathbf{v_1}$ and



$\mathbf{v_2}$ at 2D coordinates $(0, 0)$ and $(l_1, 0)$, respectively. Using Pythagoras triangle theorem we get:

$$s_1^2 = l_3^2 - h^2, \;\; s_2^2 = l_2^2 - h^2, \;\; l_1 = s_1 + s_2.$$

Solving for s_1 we get $s_1 = (l_1^2 - l_2^2 + l_3^2)/(2l_1)$, and then

$$\mathbf{v_3} = \left(s_1, (l_3^2 - s_1^2)^{1/2}\right).$$

APPENDIX C

Zipper-curve offsetting

We show that offsetting a curve in the direction of its binormal and the negative of that direction by the same amount results in two curves of the same length.

Proposition C.0.1. *Let $\gamma(s)$ be an arc-length parameterized curve, and $\tau(s)$ and $B(s)$ its torsion and binormal at s. Assume that $\tau(s) \neq 0$. Define $\gamma_{\pm}(s) := \gamma(s) \pm w\, B(s)$, where w is the offset amount. Then $\forall s$, $\|\gamma'_{+}(s)\| = \|\gamma'_{-}(s)\|$.*

Proof. By applying the Frenet-Serret formula we obtain

$$\gamma'_{\pm}(s) = T(s) \mp w\, \tau(s) N(s),$$

where $T(s), N(s)$ are the tangent and the normal of γ at s. The result immediately follows by using the polarization identity:

$$\begin{aligned} &\|T(s) + w\, \tau(s) N(s)\|^2 - \|T(s) - w\, \tau(s) N(s)\|^2 = \\ &= 4 \left\langle T(s),\, w\, \tau(s) N(s) \right\rangle = 0, \end{aligned}$$

where the last equality is due to $T(s) \perp N(s)$. □

The result above also leads to a bound on the local change of length, i.e., speed, of the offset curve. Indeed, using the triangle inequality,

$$\|\gamma_{\pm}(s)\| \leq \|T(s)\| + \|w\, \tau(s) N(s)\| = 1 + w\, \tau(s),$$

which also means that the speed is determined by the torsion $\tau(s)$. Since the zipper-curve we design usually have a helical shape, we believe that $\tau(s)$ is kept relatively small compared to a random path on the surface.

Bibliography

[3ds16] 3dsystems. ProJet CJP 660Pro. http://www.3dsystems.com/3d-printers/professional/projet-660pro, 2016. Accessed: 2016-01-18.

[AA00] E. D. Andersen and K. D. Andersen. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High Performance Optimization*, pages 197–232. Kluwer Academic Publishers, 2000. http://www.mosek.com.

[Acc16] Accuform. T-sim. http://www.t-sim.com, 2016. Accessed: 2016-01-18.

[Add18] Additively. Additively. https://www.additively.com/, 2018. Accessed: 2018-09-01.

[Agi16] Agisoft. Agisoft photoscan standard edition 1.2.2. http://www.agisoft.com/, 2016. Accessed: 2015-12-18.

[AGJ07] Marc Alexa, Steven J. Gortler, and Tao Ju, editors. *Proceedings of the Pacific Conference on Computer Graphics and Applications, Pacific Graphics 2007, Maui, Hawaii, USA, October 29 - November 2, 2007*. IEEE Computer Society, 2007.

[AIH+08] Bradley Atcheson, Ivo Ihrke, Wolfgang Heidrich, Art Tevs, Derek Bradley, Marcus A. Magnor, and Hans-Peter Seidel. Time-resolved 3D capture of non-stationary gas flows. *ACM Trans. Graph.*, 27(5), 2008.

[AKW+16] Ergun Akleman, Shenyao Ke, You Wu, Negar Kalantar, Alireza Borhani, and Jianer Chen. Construction with physical version of quad-edge data structures. *Computers & Graphics*, 58:172–183, 2016.

[AL15] Noam Aigerman and Yaron Lipman. Orbifold tutte embeddings. *ACM Trans. Graph.*, 34(6):190:1–190:12, October 2015.

[AM10] Marc Alexa and Wojciech Matusik. Reliefs as images. *ACM Trans. Graph.*, 29(4), 2010.

[AMG+18] Thomas Alderighi, Luigi Malomo, Daniela Giorgi, Nico Pietroni, Bernd Bickel, and Paolo Cignoni. Metamolds: computational design of silicone molds. *ACM Trans. Graph.*, 37(4):136:1–136:13, 2018.

[AMT+14] Byoungkwon An, Shuhei Miyashita, Michael T. Tolley, Daniel M. Aukes, Laura Meeker, Erik D. Demaine, Martin L. Demaine, Robert J. Wood, and Daniela Rus. An end-to-end approach to making self-folded 3d surface shapes by uniform heating. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014* [DBL14a], pages 1466–1473.

[ASH15] Sami Arpa, Sabine Süsstrunk, and Roger D. Hersch. High reliefs from 3d scenes. *Comput. Graph. Forum*, 34(2):253–263, 2015.

[ATG+18] Byoungkwon An, Ye Tao, Jianzhe Gu, Tingyu Cheng, Xiang 'Anthony' Chen, Xiaoxiao Zhang, Wei Zhao, Youngwook Do, Shigeo Takahashi, Hsiang-Yun Wu, Teng Zhang, and Lining Yao. Thermorph: Democratizing 4d printing of self-folding materials and interfaces. In Mandryk et al. [MHPC18], page 260.

[Bar12] Jernej Barbič. Exact corotational linear fem stiffness matrix. Technical report, University of Southern California, 2012.

[BBAM12] Amit Bermano, Ilya Baran, Marc Alexa, and Wojciech Matusik. Shadowpix: Multiple images from self shadowing. *Comput. Graph. Forum*, 31(2):593–602, 2012.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph.*, 26(3), 2007.

[BBG+13] Amit Bermano, Philipp Brüschweiler, Anselm Grundhöfer, Daisuke Iwai, Bernd Bickel, and Markus Gross. Augmenting physical avatars using projector-based illumination. *ACM Trans. Graph.*, 32(6):189:1–189:10, 2013.

[BBO+10] Bernd Bickel, Moritz Bächer, Miguel A. Otaduy, Hyunho Richard Lee, Hanspeter Pfister, Markus Gross, and Wojciech Matusik. Design and fabrication of materials with desired deformation behavior. *ACM Trans. Graph.*, 29(4):63:1–63:10, July 2010.

[BBPW04] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In Pajdla and Matas [PM04], pages 25–36.

[BCMP18] Bernd Bickel, Paolo Cignoni, Luigi Malomo, and Nico Pietroni. State of the art on stylized fabrication. *Comput. Graph. Forum*, 37(6):325–342, 2018.

[BDW14] Hrvoje Benko, Mira Dontcheva, and Daniel Wigdor, editors. *The 27th Annual ACM Symposium on User Interface Software and Technology, UIST '14, Honolulu, HI, USA, October 5-8, 2014*. ACM, 2014.

[BFH+98] Joachim M. Buhmann, Dieter W. Fellner, Marcus Held, Jens Ketterer, and Jan Puzicha. Dithered color quantization. *Comput. Graph. Forum*, 17(3):219–232, 1998.

[BFK+08] Andreas Butz, Brian D. Fisher, Antonio Krüger, Patrick Olivier, and Marc Christie, editors. *Smart Graphics, 8th International Symposium, SG 2008, Rennes, France, August 27-29, 2008. Proceedings*, volume 5166 of *Lecture Notes in Computer Science*. Springer, 2008.

[BFR17] Amit H. Bermano, Thomas A. Funkhouser, and Szymon Rusinkiewicz. State of the art in methods and representations for fabrication-aware design. *Comput. Graph. Forum*, 36(2):509–535, 2017.

[BH11] Zhe Bian and Shi-Min Hu. Preserving detailed features in digital bas-relief making. *Computer Aided Geometric Design*, 28(4):245–256, 2011.

[BHU+16] Saskia Bakker, Caroline Hummels, Brygg Ullmer, Luc Geurts, Bart Hengeveld, Daniel Saakes, and Mendel Broekhuijsen, editors. *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction, Eindhoven, The Netherlands, February 14-17, 2016*. ACM, 2016.

[BJ70] David Barnette and Ernest Jucovič. Hamiltonian circuits on 3-polytopes. *Journal of Combinatorial Theory*, 9(1):54 – 59, 1970.

[BKB+12] Ilya Baran, Philipp Keller, Derek Bradley, Stelian Coros, Wojciech Jarosz, Derek Nowrouzezahrai, and Markus Gross. Manufacturing

layered attenuators for multiple prescribed shadow images. *Comput. Graph. Forum*, 31(2):603–610, 2012.

[BKIW15] Bo Begole, Jinwoo Kim, Kori Inkpen, and Woontack Woo, editors. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*. ACM, 2015.

[BKP+10] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*. A K Peters, 2010.

[BKY99] Peter N. Belhumeur, David J. Kriegman, and Alan L. Yuille. The bas-relief ambiguity. *International Journal of Computer Vision*, 35(1):33–44, 1999.

[BLP+13] David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. Quad-mesh generation and processing: A survey. *Comput. Graph. Forum*, 32(6):51–76, 2013.

[BS08] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Trans. Vis. Comput. Graph.*, 14(1):213–230, 2008.

[BUAG12] Christopher Batty, Andres Uribe, Basile Audoly, and Eitan Grinspun. Discrete viscous sheets. *ACM Trans. Graph.*, 31(4), 2012.

[BWBSH14] Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. Spin-It: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph.*, 33(4):96:1–96:10, 2014.

[CBJ00] Hansen F. Chen, Peter N. Belhumeur, and David W. Jacobs. In search of illumination invariants. In *Proc. CVPR*, pages 1254–1261, 2000.

[CCP+05] Vincent Cheutet, Chiara Eva Catalano, Jean-Philippe Pernot, Bianca Falcidieno, Franca Giannini, and Jean-Claude Léon. 3d sketching for aesthetic design using fully free-form deformation features. *Computers & Graphics*, 29(6):916–930, 2005.

[CHM+10] Hung-Kuo Chu, Wei-Hsin Hsu, Niloy J. Mitra, Daniel Cohen-Or, Tien-Tsin Wong, and Tong-Yee Lee. Camouflage images. *ACM Trans. Graph.*, 29(3), 2010.

[CKK05] Manmohan Krishna Chandraker, Fredrik Kahl, and David J. Kriegman. Reflections on the generalized bas-relief ambiguity. In *2005*

IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA, pages 788–795, 2005.

[CKK+15] Suryansh Chandra, Axel Körner, Antiopi Koronaki, Rachelle Spiteri, Radhika Amin, Samidha Kowli, and Michael Weinstock. Computing curved-folded tessellations through straight-folding approximation. In *Proc. Symposium on Simulation for Architecture & Urban Design*, 2015.

[CMS97] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. Computer-assisted generation of bas-and high-reliefs. *J. Graphics, GPU, & Game Tools*, 2(3):15–28, 1997.

[COCSD03] Daniel Cohen-Or, Yiorgos Chrysanthou, Cláudio T. Silva, and Frédo Durand. A survey of visibility for walkthrough applications. *IEEE Trans. Vis. Comput. Graph.*, 9(3):412–431, 2003.

[Com18] Blender Online Community. Blender - a 3d modelling and rendering package. `http://www.blender.org`, 2018. Accessed: 2018-09-01.

[CPMS14] Paolo Cignoni, Nico Pietroni, Luigi Malomo, and Roberto Scopigno. Field-aligned mesh joinery. *ACM Trans. Graph.*, 33(1), 2014.

[CSLM13] Desai Chen, Pitchaya Sitthi-amorn, Justin T. Lan, and Wojciech Matusik. Computing and fabricating multiplanar models. *Comput. Graph. Forum*, 32(2):305–315, 2013.

[Cut15] Cuttlefish. Cuttlefish: 3D printing pipeline. `https://www.cuttlefish.de/`, 2015. Accessed: 2015-03-01.

[CW01] Umberto Cugini and Michael J. Wozny, editors. *From Geometric Modeling to Shape Modeling, IFIP TC5 WG5.2 Seventh Workshop on Geometric Modeling: Fundamentals and Applications, October 2-4, 2000, Parma, Italy*, volume 208 of *IFIP Conference Proceedings*. Kluwer, 2001.

[CWT08] Chih-Hsing Chu, Charlie C. L. Wang, and Chi-Rung Tsai. Computer aided geometric design of strip using developable Bézier patches. *Computers in Industry*, 59(6):601–611, 2008.

[CZXZ14] Xiang Chen, Changxi Zheng, Weiwei Xu, and Kun Zhou. An asymptotic numerical method for inverse elastic shape design. *ACM Trans. Graph.*, 33(4), 2014.

[dav15] DAVID structured light scanner. `http://www.david-3d.com/en/`, 2015. Accessed: 2015-03-01.

[DBL04a] *12th Pacific Conference on Computer Graphics and Applications (PG 2004), 6-8 October 2004, Seoul, Korea*. IEEE Computer Society, 2004.

[DBL04b] *2004 Computer Graphics International (CGI 2004), 16-19 June 2004, Crete, Greece*. IEEE Computer Society, 2004.

[DBL07] *2007 International Conference on Shape Modeling and Applications (SMI 2007), 13-15 June 2007, Lyon, France*. IEEE Computer Society, 2007.

[DBL10] *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry, Winnipeg, Manitoba, Canada, August 9-11, 2010*, 2010.

[DBL14a] *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*. IEEE, 2014.

[DBL14b] *Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '14, Vancouver, Canada, August 10-14, 2014, Studio*. ACM, 2014.

[DDP02] Frédo Durand, George Drettakis, and Claude Puech. The 3D visibility complex. *ACM Trans. Graph.*, 21(2):176–206, 2002.

[DFS05] Neil A. Dodgson, Michael S. Floater, and Malcolm A. Sabin, editors. *Advances in Multiresolution for Geometric Modelling*. Springer, 2005.

[DM06] Douglas DeCarlo and Lee Markosian, editors. *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering 2006, Annecy, France, June 5-7, 2006*. ACM, 2006.

[DPW+14] Mario Deuss, Daniele Panozzo, Emily Whiting, Yang Liu, Philippe Block, Olga Sorkine-Hornung, and Mark Pauly. Assembling self-supporting structures. *ACM Trans. Graph.*, 33(6), 2014.

[EB14] Essex Edwards and Robert Bridson. Detailed water with coarse grids: combining surface meshes and adaptive discontinuous Galerkin. *ACM Trans. Graph.*, 33(4), 2014.

[EF12] Leah Epstein and Paolo Ferragina, editors. *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, volume 7501 of *Lecture Notes in Computer Science*. Springer, 2012.

[EG04] David Eppstein and M. Gopi. Single-strip triangulation of manifolds with arbitrary topology. In *Proc. Symp. Computational Geometry*, pages 455–456, 2004.

[EG13] Peter Eisert and André Gagalowicz, editors. *6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications, MIRAGE '13, Berlin, Germany - June 06 - 07, 2013*. ACM, 2013.

[Elb10] Gershon Elber. Ortho-pictures: 3D objects from independent 2D data sets. In Cristiano Ceccato et al., editors, *Advances in Architectural Geometry 2010*, pages 175–192. Springer, 2010.

[EPRG13] Andreas Ernst, Anton Papst, Tobias Ruf, and Jens-Uwe Garbas. Check my chart: a robust color chart tracker for colorimetric camera calibration. In Eisert and Gagalowicz [EG13], pages 5:1–5:8.

[ESI16] ESI. ESI Pam-form. `https://www.esi-group.com/software-solutions/virtual-manufacturing/composites/solutions-plastics-trims`, 2016. Accessed: 2016-01-18.

[FBR+17] Amélie Fondevilla, Adrien Bousseau, Damien Rohmer, Stefanie Hahmann, and Marie-Paule Cani. Patterns from photograph: Reverse-engineering developable products. *Computers & Graphics*, 66:4–13, 2017.

[FH93] Dieter W. Fellner and Christoph Helmberg. Robust rendering of general ellipses and elliptical arcs. *ACM Transactions on Graphics*, 12(3):251–276, July 1993.

[FH05] Michael S. Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In Dodgson et al. [DFS05], pages 157–186.

[Fou] Fab Foundation. Fab foundation. `http://www.fabfoundation.org/`. Accessed: 2018-09-01.

[FP06] John W. Finnegan and Hanspeter Pfister, editors. *33. International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2006, Boston, Massachusetts, USA, July 30 - August 3, 2006, Sketches*. ACM, 2006.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 15–22. ACM, ACM Press / ACM SIGGRAPH, 2001.

[FSY+15] Chi-Wing Fu, Peng Song, Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman, and Daniel Cohen-Or. Computational interlocking furniture assembly. *ACM Trans. Graph.*, 34(4):91, 2015.

[FUJ16] FUJIFILM. Thermoforming ink. `http://www.fujifilm.com.au/powerofinkjet/applications/thermoforming`, 2016. Accessed: 2016-04-19.

[FUM+10] Yohsuke Furuta, Nobuyuki Umetani, Jun Mitani, Takeo Igarashi, and Yukio Fukui. A film balloon design system integrated with shell element simulation. In Lensch and Seipel [LS10], pages 33–36.

[FvDF+93] James D. Foley, Andries van Dam, Stephen K. Feiner, John F. Hughes, and R. Phillips. *Introduction to Computer Graphics*. Addison-Wesley, 1993.

[GCMT14] Damien Gauge, Stelian Coros, Sandro Mani, and Bernhard Thomaszewski. Interactive design of modular tensegrity characters. In Koltun and Sifakis [KS14], pages 131–138.

[GMB17] Ruslan Guseinov, Eder Miguel, and Bernd Bickel. Curveups: shaping objects from flat plates with tension-actuated curvature. *ACM Trans. Graph.*, 36(4):64:1–64:12, 2017.

[Gop04] Meenakshisundaram Gopi. Controllable single-strip generation for triangulated surfaces. In *12th Pacific Conference on Computer Graphics and Applications (PG 2004), 6-8 October 2004, Seoul, Korea* [DBL04a], pages 61–69.

[GSA12] Carlo Giovannella, Demetrios G. Sampson, and Ignacio Aedo, editors. *12th IEEE International Conference on Advanced Learning Technologies, ICALT 2012, Rome, Italy, July 4-6, 2012*. IEEE Computer Society, 2012.

[GSD+14] Akash Garg, Andrew O. Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. Wire mesh design. *ACM Trans. Graph.*, 33(4):66:1–66:12, 2014.

[HAM14] Philipp Herholz, Marc Alexa, and Wojciech Matusik. Turning free-form surfaces into manufacturable components. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '14, Vancouver, Canada, August 10-14, 2014, Studio* [DBL14b], page 11:1.

[HB04] Wolfgang Heidrich and Ravin Balakrishnan, editors. *Proceedings of the Graphics Interface 2004 Conference, May 17-19, 2004, London, Ontario, Canada*, volume 62 of *ACM International Conference Proceeding Series*. Canadian Human-Computer Communications Society, 2004.

[HBA12] Kristian Hildebrand, Bernd Bickel, and Marc Alexa. crdbrd: Shape fabrication by sliding planar slices. *Comput. Graph. Forum*, 31(2):583–592, 2012.

[HBA13] Kristian Hildebrand, Bernd Bickel, and Marc Alexa. Orthogonal slicing for additive manufacturing. *Computers & Graphics*, 37(6):669–675, 2013.

[Hil98] R. Hill. *The Mathematical Theory of Plasticity*. Oxford classic texts in the physical sciences. Clarendon Press, 1998.

[HKL18] Yue Hao, Yun-hyeong Kim, and Jyh-Ming Lien. Synthesis of fast and collision-free folding of polyhedral nets. In *Proceedings of the 2Nd ACM Symposium on Computational Fabrication*, SCF '18, pages 2:1–2:10, New York, NY, USA, 2018. ACM.

[HL14] Jean Hergel and Sylvain Lefebvre. Clean color: Improving multi-filament 3D prints. *Computer Graphics Forum*, 33(2), 2014.

[HLS07] Kai Hormann, Bruno Lévy, and Alla Sheffer. Mesh parameterization: theory and practice. In *ACM SIGGRAPH Courses*, 2007.

[HLZC14] Ruizhen Hu, Honghua Li, Hao Zhang, and Daniel Cohen-Or. Approximate pyramidal shape decomposition. *ACM Trans. Graph.*, 33(6):213:1–213:12, 2014.

[HM08] Eric Haines and Morgan McGuire, editors. *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling, Stony Brook, New York, USA, June 2-4, 2008*. ACM, 2008.

[HMA15] Philipp Herholz, Wojciech Matusik, and Marc Alexa. Approximating free-form geometry with height fields for manufacturing. *Comput. Graph. Forum*, 34(2):239–251, 2015.

[HS01] John F. Hughes and Carlo H. Séquin, editors. *Proceedings of the 2001 Symposium on Interactive 3D Graphics, SI3D 2001, Chapel Hill, NC, USA, March 26-29, 2001*. ACM, 2001.

[HSV05] Toon Huysmans, Jan Sijbers, and Brigitte Verdonk. Parametrization of tubular surfaces on the cylinder. *Journal of WSCG*, 13(3):97–104, 2005.

[HTG14] Gur Harary, Ayellet Tal, and Eitan Grinspun. Context-based coherent surface completion. *ACM Trans. Graph.*, 33(1):5, 2014.

[Hud14] Scott E. Hudson. Printing teddy bears: a technique for 3d printing of soft interactive objects. In Jones et al. [JPSG14], pages 459–468.

[HWS+16] Jhen-Yao Hong, Der-Lor Way, Zen-Chung Shih, Wen-Kai Tai, and Chin-Chen Chang. Inner engraving for the creation of a balanced LEGO sculpture. *The Visual Computer*, 32(5):569–578, 2016.

[II08] Yuki Igarashi and Takeo Igarashi. Pillow: Interactive flattening of a 3D model for plush toy design. In *Proc. Smart Graphics*, 2008.

[IIM12] Yuki Igarashi, Takeo Igarashi, and Jun Mitani. Beady: Interactive beadwork design and construction. *ACM Trans. Graph.*, 31(4):49:1–49:9, 2012.

[IIS09] Yuki Igarashi, Takeo Igarashi, and Hiromasa Suzuki. Interactive cover design considering physical constraints. *Comput. Graph. Forum*, 28(7):1965–1973, 2009.

[Inc18] Vaquform Inc. Vaquform, 2018.

[Isr11] Jacob N Israelachvili. *Intermolecular and surface forces: revised third edition*. Academic press, 2011.

[JHR+15] Amaury Jung, Stefanie Hahmann, Damien Rohmer, Antoine Bégault, Laurence Boissieux, and Marie-Paule Cani. Sketching folds: Developable surfaces from non-planar silhouettes. *ACM Trans. Graph.*, 34(5):155:1–155:12, 2015.

[JKS05] Dan Julius, Vladislav Kraevoy, and Alla Sheffer. D-Charts: Quasi-developable mesh segmentation. *Comput. Graph. Forum*, 24(3):581–590, 2005.

[JLYL14] Conrado R. Ruiz Jr., Sang N. Le, Jinze Yu, and Kok-Lim Low. Multi-style paper pop-up designs from 3d models. *Comput. Graph. Forum*, 33(2):487–496, 2014.

[JMS14] Zhongping Ji, Weiyin Ma, and Xianfang Sun. Bas-relief modeling from normal images with intuitive styles. *IEEE Trans. Vis. Comput. Graph.*, 20(5):675–685, 2014.

[JPSG14] Matt Jones, Philippe A. Palanque, Albrecht Schmidt, and Tovi Grossman, editors. *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014*. ACM, 2014.

[JRW95] Daniel J Jobson, Zia-ur Rahman, and Glenn A Woodell. Retinex image processing: Improved fidelity to direct visual observation. In *Proceedings of the IS&T Fourth Color Imaging Conference: Color Science, Systems, and Applications*, volume 4, pages 124–125. The Society for Imaging Science and Technology, 1995.

[Kar00] Daniel Kartch. *Efficient Rendering and Compression for Full-Parallax Computer-Generated Holographic Stereograms*. PhD thesis, Cornell University, 2000.

[KBV92] K. Kouba, O. Bartos, and J. Vlachopoulos. Computer simulation of thermoforming in complex shapes. *Polymer Engineering and Science*, 32(10):699–704, 1992.

[KCD+16] Mina Konakovic, Keenan Crane, Bailin Deng, Sofien Bouaziz, Daniel Piker, and Mark Pauly. Beyond developable: computational design and fabrication with auxetic materials. *ACM Trans. Graph.*, 35(4), 2016.

[KCPS15] Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. Stripe patterns on surfaces. *ACM Trans. Graph.*, 34(4), 2015.

[KDL+16] Jofish Kaye, Allison Druin, Cliff Lampe, Dan Morris, and Juan Pablo Hourcade, editors. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016*. ACM, 2016.

[KFC+08] Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J. Mitra, Alla Sheffer, and Helmut Pottmann. Curved folding. *ACM Trans. Graph.*, 27(3), 2008.

[KG02] Simon Kolmianič and Nikola Guid. From geometric modeling to shape modeling. In *Proc. Workshop on Geometric Modeling: Fundamentals and Applications*, pages 35–46, 2002.

[KGL16] Shahar Z. Kovalsky, Meirav Galun, and Yaron Lipman. Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph.*, 35(4):134, 2016.

[KGVM97] B.L. Koziey, M.O. Ghafur, J. Vlachopoulos, and F.A. Mirza. Computer simulation of thermoforming. In D. Bhattacharyya, editor, *Composite Sheet Forming*, volume 11 of *Composite Materials Series*, chapter 3, pages 75 – 89. Elsevier, 1997.

[KLC+15] Ming-Hsun Kuo, You-En Lin, Hung-Kuo Chu, Ruen-Rone Lee, and Yong-Liang Yang. Pixel2brick: Constructing brick sculptures from pixel art. *Comput. Graph. Forum*, 34(7):339–348, 2015.

[Kle09] P. Klein. *Fundamentals of Plastics Thermoforming*. Synthesis lectures on materials engineeering. Morgan & Claypool, 2009.

[KLS13] A. Kuzmin, M. Luisier, and O. Schenk. Fast methods for computing selected elements of the greens function in massively parallel nanoelectronic device simulations. In F. Wolf, B. Mohr, and D. Mey, editors, *Euro-Par 2013 Parallel Processing*, volume 8097 of *Lecture Notes in Computer Science*, pages 533–544. Springer Berlin Heidelberg, 2013.

[KLST11] Michael Kolomenkin, George Leifman, Ilan Shimshoni, and Ayellet Tal. Reconstruction of relief objects from line drawings. In *Proc. CVPR*, pages 993–1000, 2011.

[KMM17] Martin Kilian, Aron Monszpart, and Niloy J. Mitra. String actuated curved folded surfaces. *ACM Trans. Graph.*, 36(3):25:1–25:13, 2017.

[KNP11] Felix Kälberer, Matthias Nieser, and Konrad Polthier. Stripe parameterization of tubular surfaces. In *Topological Methods in Data Analysis and Visualization*, pages 13–26. Springer, 2011.

[KPCP18] Mina Konakovic-Lukovic, Julian Panetta, Keenan Crane, and Mark Pauly. Rapid deployment of curved surfaces via programmable auxetics. *ACM Trans. Graph.*, 37(4):106:1–106:13, 2018.

[KS14] Vladlen Koltun and Eftychios Sifakis, editors. *The Eurographics / ACM SIGGRAPH Symposium on Computer Animation, SCA 2014, Copenhagen, Denmark, 2014*. Eurographics Association, 2014.

[KSS97] Leif Kobbelt, Marc Stamminger, and Hans-Peter Seidel. Using subdivision on hierarchical data to reconstruct radiosity distribution. *Comput. Graph. Forum*, 16(3):347–356, 1997.

[KTB+09] Jens Kerber, Art Tevs, Alexander G. Belyaev, Rhaleb Zayer, and Hans-Peter Seidel. Feature sensitive bas relief generation. In *Shape Modeling International*, pages 148–154, 2009.

[KTM16] Takuya Kozaki, Hiroshi Tedenuma, and Takashi Maekawa. Automatic generation of LEGO building instructions from multiple photographic images of real objects. *Computer-Aided Design*, 70:13–22, 2016.

[Lan02] H. Landis. Global illumination in production. ACM SIGGRAPH 2002 Course #16 Notes, July 2002.

[LAPS17] Marco Livesu, Marco Attene, Giuseppe Patanè, and Michela Spagnuolo. Explicit cylindrical maps for general tubular shapes. *Computer-Aided Design*, 2017. Accepted for publication.

[LBRM12] Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. Chopper: Partitioning models into 3D-printable parts. *ACM Trans. Graph.*, 31(6), 2012.

[LDD+10] Anna Lubiw, Erik D. Demaine, Martin L. Demaine, Arlo Shallit, and Jonah Shallit. Zipper unfoldings of polyhedral complexes. In *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry, Winnipeg, Manitoba, Canada, August 9-11, 2010* [DBL10], pages 219–222.

[LFL09] Kui-Yip Lo, Chi-Wing Fu, and Hongwei Li. 3d polyomino puzzle. *ACM Trans. Graph.*, 28(5):157:1–157:8, 2009.

[LFTG97] Eric P. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *Proc. SIGGRAPH '97*, volume 31, pages 117–126, 1997.

[LHG15] Celine Latulipe, Bjoern Hartmann, and Tovi Grossman, editors. *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST 2015, Charlotte, NC, USA, November 8-11, 2015*. ACM, 2015.

[LHN05] Sylvain Lefebvre, Samuel Hornus, and Fabrice Neyret. Texture sprites: Texture elements splatted on surfaces. In *Proc. ACM I3D*, 2005.

[LJGH11] Xian-Ying Li, Tao Ju, Yan Gu, and Shi-Min Hu. A geometric study of v-style pop-ups: theories and algorithms. *ACM Trans. Graph.*, 30(4):98:1–98:10, 2011.

[LLH09] Y.-J. Liu, Y.-K. Lai, and S. M. Hu. Stripification of free-form surfaces with global error bounds for developable approximation. *IEEE Trans. Automation Science and Engineering*, 6(4):700–709, 2009.

[LLL+14] Sang N. Le, Su Jun Leow, Tuong-Vu Le-Nguyen, Conrado R. Ruiz Jr., and Kok-Lim Low. Surface and contour-preserving origamic architecture paper pop-ups. *IEEE Trans. Vis. Comput. Graph.*, 20(2):276–288, 2014.

[Lou] Yvon Le Lous. Report on the first eurographics workshop on visualization in scientific computing. *Computer Graphics Forum*, 9(5):371–372.

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 131–144, New York, 2000. ACM, ACM Press / ACM SIGGRAPH.

[LPD13] Thomas Lindemeier, Sören Pirk, and Oliver Deussen. Image stylization with a painting machine using semantic hints. *Computers & Graphics*, 37(5), 2013.

[LS10] Hendrik P. A. Lensch and Stefan Seipel, editors. *Eurographics 2010 - Short Papers, Norrköping, Sweden, May 3-7, 2010*. Eurographics Association, 2010.

[LSH+10] Xian-Ying Li, Chao-Hui Shen, Shi-Sheng Huang, Tao Ju, and Shi-Min Hu. Popup: automatic paper architectures from 3d models. *ACM Trans. Graph.*, 29(4):111:1–111:9, 2010.

[Ltd16] Mcor Technologies Ltd. Mcor Technologies Ltd. `http://mcortechnologies.com/`, 2016. Accessed: 2016-01-18.

[LWN+09] Peter Lincoln, Greg Welch, Andrew Nashel, Adrian Ilie, Andrei State, and Henry Fuchs. Animatronic shader lamps avatars. In *Proc. ISMAR*, pages 27–33, 2009.

[LXG10] Ali Lasemi, Deyi Xue, and Peihua Gu. Recent development in CNC machining of freeform surfaces: A state-of-the-art review. *Computer-Aided Design*, 42(7):641–654, 2010.

[LYH+15] Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. Legolization: optimizing LEGO designs. *ACM Trans. Graph.*, 34(6):222:1–222:12, 2015.

[LZX+08] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. A local/global approach to mesh parameterization. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1495–1504, 2008.

[mak15] Makerbot replicator. `http://www.makerbot.com/`, 2015. Accessed: 2015-03-01.

[MAN+16] James McCann, Lea Albaugh, Vidya Narayanan, April Grow, Wojciech Matusik, Jennifer Mankoff, and Jessica K. Hodgins. A compiler for 3d machine knitting. *ACM Trans. Graph.*, 35(4):49:1–49:11, 2016.

[MAP+12] Paolo Menesatti, Claudio Angelini, Federico Pallottino, Francesca Antonucci, Jacopo Aguzzi, and Corrado Costa. Rgb color calibration for quantitative image analysis: the "3d thin-plate spline" warping approach. *Sensors*, 12(12):7063–7079, May 2012.

[MAWS15] Ali Mahdavi-Amiri, Philip Whittingham, and Faramarz Samavati. Cover-it: an interactive system for covering 3D prints. In *Proc. Graphics Interface*, pages 73–80, 2015.

[May18] Mayku. Formbox, 2018.

[MBL12] Rob Miller, Hrvoje Benko, and Celine Latulipe, editors. *The 25th Annual ACM Symposium on User Interface Software and Technology, UIST '12, Cambridge, MA, USA, October 7-10, 2012 - Adjunct Volume*. ACM, 2012.

[MBW16] Bewketu Gizachew Mekonnen, Glen Bright, and Anthony Walker. A study on state of the art technology of laminated object manufacturing (lom). In Dipak Kumar Mandal and Chanan Singh Syan, editors, *CAD/CAM, Robotics and Factories of the Future*, pages 207–216, New Delhi, 2016. Springer India.

[MCK08] Tobias Martin, Elaine Cohen, and Mike Kirby. Volumetric parameterization and trivariate b-spline fitting using harmonic functions. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling, Stony Brook, New York, USA, June 2-4, 2008*, pages 269–280, 2008.

[MDBI15] Bart Van Mieghem, Frederik Desplentere, Albert Van Bael, and Jan Ivens. Improvements in thermoforming simulation by use of 3D digital image correlation. *Express Polymer Letters*, 9(2), 2015.

[MDSH03] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H.Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, 2003.

[MFL+17a] Gloria Mark, Susan R. Fussell, Cliff Lampe, m. c. schraefel, Juan Pablo Hourcade, Caroline Appert, and Daniel Wigdor, editors. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017*. ACM, 2017.

[MFL+17b] Gloria Mark, Susan R. Fussell, Cliff Lampe, m. c. schraefel, Juan Pablo Hourcade, Caroline Appert, and Daniel Wigdor, editors. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017, Extended Abstracts*. ACM, 2017.

[MG04] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proc. Graphics Interface*, pages 239–246, 2004.

[MGE07] Fady Massarwi, Craig Gotsman, and Gershon Elber. Papercraft models using generalized cylinders. In *Proc. Pacific Graphics*, 2007.

[MHPC18] Regan L. Mandryk, Mark Hancock, Mark Perry, and Anna L. Cox, editors. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*. ACM, 2018.

[MI06] Yuki Mori and Takeo Igarashi. Pillow: interactive pattern design for stuffed animals. In Finnegan and Pfister [FP06], page 74.

[MI07] Yuki Mori and Takeo Igarashi. Plushie: an interactive design system for plush toys. *ACM Trans. Graph.*, 26(3), 2007.

[MLB16] Eder Miguel, Mathias Lepoutre, and Bernd Bickel. Computational design of stable planar-rod structures. *ACM Trans. Graph.*, 35(4):86, 2016.

[MLS+ar] Alessandro Muntoni, Marco Livesu, Riccardo Scateni, Alla Sheffer, and Daniele Panozzo. Axis-aligned height-field block decomposition of 3d shapes. *ACM Transactions on Graphics*, (to appear).

[MP09] N. J. Mitra and M. Pauly. Shadow art. *ACM Trans. Graph.*, 28(5), 2009.

[MPBC16] Luigi Malomo, Nico Pietroni, Bernd Bickel, and Paolo Cignoni. Flexmolds: automatic design of flexible shells for molding. *ACM Trans. Graph.*, 35(6):223:1–223:12, 2016.

[MQW01] Kevin T. McDonnell, Hong Qin, and Robert A. Wlodarczyk. Virtual clay: a real-time sculpting system with haptic toolkits. In Hughes and Séquin [HS01], pages 179–190.

[MS04] Jun Mitani and Hiromasa Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Trans. Graph.*, 23(3):259–263, 2004.

[MS07] Sara McMains and Peter-Pike Sloan, editors. *34. International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2007, San Diego, California, USA, August 5-9, 2007, Courses*. ACM, 2007.

[MSM11] James McCrae, Karan Singh, and Niloy J. Mitra. Slices: a shape-proxy based on planar sections. *ACM Trans. Graph.*, 30(6):168:1–168:12, 2011.

[MUS14] James McCrae, Nobuyuki Umetani, and Karan Singh. FlatFitFab: interactive modeling with planar sections. In *Proc. UIST*, pages 13–22, 2014.

[MZ12] Ashish Myles and Denis Zorin. Global parametrization by incremental flattening. *ACM Trans. Graph.*, 31(4):109:1–109:11, 2012.

[MZS+11] Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph.*, 30(4):37:1–37:12, 2011.

[NAH+18] Vidya Narayanan, Lea Albaugh, Jessica Hodgins, Stelian Coros, and James Mccann. Automatic machine knitting of 3d meshes. *ACM Trans. Graph.*, 37(3):35:1–35:15, August 2018.

[NAI+18] Kazutaka Nakashima, Thomas Auzinger, Emmanuel Iarussi, Ran Zhang, Takeo Igarashi, and Bernd Bickel. Corecavity: interactive shell decomposition for fabrication with two-piece rigid molds. *ACM Trans. Graph.*, 37(4):135:1–135:13, 2018.

[Nak84] M. Nakanishi. Continuous transcriptions of a pattern onto an article, March 1984. US Patent 4,436,571.

[NTD90] H. F. Nied, C. A. Taylor, and H. G. Delorenzi. Three-dimensional finite element simulation of thermoforming. *Polymer Engineering and Science*, 30(20):1314–1322, 1990.

[OBH02] James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph.*, 21(3):291–294, 2002.

[O'R15] Joseph O'Rourke. Spiral unfoldings of convex polyhedra. *arXiv preprint arXiv:1509.00321*, 2015.

[OS13] Miguel A. Otaduy and Olga Sorkine, editors. *Eurographics 2013 - Short Papers Proceedings, Girona, Spain, May 6-10, 2013*. Eurographics Association, 2013.

[PAB+16] Kazim Pal, Nicole Avery, Pete Boston, Alberto Campagnolo, Caroline De Stefani, Helen Matheson-Pollock, Daniele Panozzo, Matthew Payne, **Schüller, Christian**, Chris Sanderson, Chris Scott, Philippa Smith, Rachael Smither, Olga Sorkine-Hornung, Ann Stewart, Emma Stewart, Patricia Stewart, Melissa Terras, Bernadette Walsh, Laurence Ward, Liz Yamada, and Tim Weyrich. Digitally reconstructing the great parchment book: 3D recovery of fire-damaged historical documents. *Digital Scholarship in the Humanities*, December 2016.

[PBCW07] Helmut Pottmann, Sigrid Brell-Cokcan, and Johannes Wallner. Discrete surfaces for architectural design. In Patrick Chenin, Tom Lyche, and Larry L. Schumaker, editors, *Curves and Surface Design: Avignon 2006*, pages 213–234. Nashboro Press, 2007.

[PBSH13] Daniele Panozzo, Philippe Block, and Olga Sorkine-Hornung. Designing unreinforced masonry models. *ACM Trans. Graph.*, 32(4):91:1–91:12, 2013.

[PDP+15] Daniele Panozzo, Olga Diamanti, Sylvain Paris, Marco Tarini, Evgeni Sorkine, and Olga Sorkine-Hornung. Texture mapping real-world objects with hydrographics. *Comput. Graph. Forum*, 34(5), 2015.

[PDRhK18] Patrick Paczkowski, Julie Dorsey, Holly Rushmeier, and Min hyuk Kim. Papercraft3d: Paper-based 3d modeling and scene fabrication. *IEEE transactions on visualization and computer graphics*, 2018.

[Ped11] H.K. Pedersen. Methods for creating developable surfaces, July 14 2011. US Patent App. 13/005,384.

[PFGL08] Jean-Philippe Pernot, Bianca Falcidieno, Franca Giannini, and J.-C. Léon. Incorporating free-form features in aesthetic and engineering product design: State-of-the-art report. *Computers in Industry*, 59(6):626–637, 2008.

[PHB16] Zherong Pan, Jin Huang, and Hujun Bao. Modelling developable ribbons using ruling bending coordinates. Technical report, 2016. `http://arxiv.org/abs/1603.04060`.

[PJJSH15] Romain Prévost, Alec Jacobson, Wojciech Jarosz, and Olga Sorkine-Hornung. Large-scale painting of photographs by interactive optimization. *Computers & Graphics*, 2015. In press.

[PLK+06] S. W. Park, L. Linsen, O. Kreylos, J. D. Owens, and B. Hamann. Discrete sibson interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):243–253, March/April 2006.

[PM04] Tomás Pajdla and Jiri Matas, editors. *Computer Vision - ECCV 2004, 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part IV*, volume 3024 of *Lecture Notes in Computer Science*. Springer, 2004.

[PMHM15] Huaishu Peng, Jennifer Mankoff, Scott E. Hudson, and James McCann. A layered fabric 3d printer for soft interactive objects. In Begole et al. [BKIW15], pages 1789–1798.

[pro15] HydroGraphix Print Film: Instruction and Storage Information. `http://www.prostreetgraphix.com/instructions/`, 2015. Accessed: 2015-03-01.

[PS06] Hans Pedersen and Karan Singh. Organic labyrinths and mazes. In DeCarlo and Markosian [DM06], pages 79–86.

[PtP+14] Kazim Pal, **Christian Schüller**, Daniele Panozzo, Olga Sorkine-Hornung, and Tim Weyrich. Content-aware surface parameterization for interactive restoration of historical documents. *To appear in Computer Graphics Forum (Proc. Eurographics)*, 33(2), 2014.

[PTV+17] Nico Pietroni, Marco Tarini, Amir Vaxman, Daniele Panozzo, and Paolo Cignoni. Position-based tensegrity design. *ACM Trans. Graph.*, 36(6):172:1–172:14, 2017.

[PVL+05] Fabio Pellacini, Kiril Vidimče, Aaron Lefohn, Alex Mohr, Mark Leone, and John Warren. Lpics: a hybrid hardware-accelerated relighting engine for computer cinematography. *ACM Transactions on Graphics*, 24(3):464–470, August 2005.

[PW96] Frederic I. Parke and Keith Waters. *Computer Facial Animation*. A. K. Peters, 1996.

[PWLSH13] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make It Stand: Balancing shapes for 3D fabrication. *ACM Trans. Graph.*, 32(4):81:1–81:10, 2013.

[RA15] Ronald Richter and Marc Alexa. Beam meshes. *Computers & Graphics*, 53:28–36, 2015.

[RAD12] Alec R. Rivers, Andrew Adams, and Frédo Durand. Sculpting by numbers. *ACM Trans. Graph.*, 31(6):157:1–157:7, 2012.

[Rav06] B. Ravi. *Metal casting: computer-aided design and analysis*. Prentice-Hall, 2006.

[RBK+13] Olivier Rouiller, Bernd Bickel, Jan Kautz, Wojciech Matusik, and Marc Alexa. 3D-printing spatially varying BRDFs. *IEEE Computer Graphics and Applications*, 33(6), 2013.

[RCM+14] Tim Reiner, Nathan Carr, Radomír Měch, Ondřej Šťava, Carsten Dachsbacher, and Gavin Miller. Dual-color mixing for fused deposition modeling printers. *Computer Graphics Forum*, 33(2), 2014.

[Rhe16] Rheoware. Rheoware simulation. `http://www.blowmolding-thermoforming-simulation.com`, 2016. Accessed: 2016-01-18.

[Rif13] Jeremy Rifkin. *The third industrial revolution: how lateral power is transforming energy, the economy, and the world*. Palgrave Macmillan, 2013.

[RLL+06] Nicolas Ray, Wan-Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. Periodic global parameterization. *ACM Trans. Graph.*, 25(4), 2006.

[Ros99] Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Trans. Vis. Comput. Graph.*, 5(1):47–61, 1999.

[RPPSH17] Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. Scalable locally injective mappings. *ACM Trans. Graph.*, 36(2):16:1–16:16, 2017.

[RRP00] Holly E Rushmeier, Bernice E Rogowitz, and Christine Piatko. Perceptual issues in substituting texture for geometry. In *Electronic Imaging*. International Society for Optics and Photonics, 2000.

[RSW+07] Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paule Cani, and Boris Thibert. Developable surfaces from arbitrary sketched boundaries. In *Proc. Symp. Geom. Processing*, pages 163–172, 2007.

[RWLB01] Ramesh Raskar, Greg Welch, Kok-lim Low, and Deepak Bandyopadhyay. Shader lamps: Animating real objects with image-based illumination. In *Proc. EG Workshop on Rendering*, 2001.

[SA07] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proc. Symposium on Geometry Processing*, pages 109–116, 2007.

[SACO04] Andrei Sharf, Marc Alexa, and Daniel Cohen-Or. Context-based surface completion. *ACM Trans. Graph.*, 23(3):878–887, 2004.

[SB04] Jack Snoeyink and Jean-Daniel Boissonnat, editors. *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*. ACM, 2004.

[SBG15] Holly Samuelson, Shajay Bhooshan, and Rhys Goldstein, editors. *Proceedings of the Symposium on Simulation for Architecture & Urban Design, SimAUD 2015, part of the 2015 Spring Simulation Multiconference, SpringSim '15, Alexandria, VA, USA, April 12-15, 2015*. SCS/ACM, 2015.

[SBHH99] Peter M. A. Sloot, Marian Bubak, Alfons G. Hoekstra, and Louis O. Hertzberger, editors. *High-Performance Computing and Networking, 7th International Conference, HPCN Europe 1999, Amsterdam, The Netherlands, April 12-14, 1999, Proceedings*, volume 1593 of *Lecture Notes in Computer Science*. Springer, 1999.

[SBM16] Ariel Shamir, Bernd Bickel, and Wojciech Matusik. Computational tools for 3D printing. In *ACM SIGGRAPH Courses*, pages 9:1–9:34, 2016.

[SBR08] Olaf Schenk, Matthias Bollhöfer, and Rudolf A. Römer. On large-scale diagonalization techniques for the anderson model of localization. *SIAM Rev.*, 50(1):91–112, February 2008.

[SBS07] Wenhao Song, Alexander G. Belyaev, and Hans-Peter Seidel. Automatic generation of bas-reliefs from 3d shapes. In *Shape Modeling International*, pages 211–214, 2007.

[SCGT15] Mélina Skouras, Stelian Coros, Eitan Grinspun, and Bernhard Thomaszewski. Interactive surface design with interlocking elements. *ACM Trans. Graph.*, 34(6):224, 2015.

[SCOL+04] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proc. Symposium on Geometry Processing*, pages 175–184, 2004.

[SDW+16] Peng Song, Bailin Deng, Ziqi Wang, Zhichao Dong, Wei Li, Chi-Wing Fu, and Ligang Liu. Cofifab: coarse-to-fine fabrication of large 3d objects. *ACM Trans. Graph.*, 35(4):45, 2016.

[SE07] Guy Sela and Gershon Elber. Generation of view dependent models using free form deformation. *The Visual Computer*, 23(3):219–229, 2007.

[Sei93] Hans-Peter Seidel. Polar forms for geometrically continuous spline curves of arbitrary degree. *ACM Transactions on Graphics*, 12(1):1–34, January 1993.

[Séq12] Carlo H. Séquin. Prototyping dissection puzzles with layered manufacturing. 2012.

[SF00] Yusaku Sako and Kikuo Fujimura. Shape similarity by homotropic deformation. *The Visual Computer*, 16(1):47–61, 2000.

[SFC12] Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. Recursive interlocking puzzles. *ACM Trans. Graph.*, 31(6):128:1–128:10, 2012.

[SFLF15] Peng Song, Zhongqi Fu, Ligang Liu, and Chi-Wing Fu. Printing 3d objects with interlocking parts. *Computer Aided Geometric Design*, 35-36:137–148, 2015.

[SG04] Olaf Schenk and Klaus Gärtner. Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Comp. Syst.*, 20(3):475–487, 2004.

[SGC18] Oded Stein, Eitan Grinspun, and Keenan Crane. Developability of triangle meshes. *ACM Trans. Graph.*, 37(4):77:1–77:14, 2018.

[SGF99] Olaf Schenk, Klaus Gärtner, and Wolfgang Fichtner. Scalable parallel sparse factorization with left-right looking strategy on shared memory multoprocessors. In Sloot et al. [SBHH99], pages 221–230.

[SGSH02] Pedro V. Sander, Steven J. Gortler, John Snyder, and Hugues Hoppe. Signal-specialized parametrization. In *Proc. EGSR*, pages 87–98, 2002.

[SH98] Juan C. Simo and Thomas J. R. Hughes. *Computational inelasticity*. Interdisciplinary applied mathematics. Springer, 1998.

[Sha08] Ariel Shamir. A survey on mesh segmentation techniques. *Comput. Graph. Forum*, 27(6):1539–1556, 2008.

[Sha18] ShaperOrigin. Shaperorigin, 2018.

[She96] Jonathan Richard Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. 1996.

[Sif12] Eftychios D. Sifakis. FEM simulation of 3D deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH Courses*, 2012.

[SJP+13] Leonardo Sacht, Alec Jacobson, Daniele Panozzo, **Christian Schüller**, and Olga Sorkine-Hornung. Consistent volumetric discretizations inside self-intersecting surfaces. *Computer Graphics Forum (proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing)*, 32(5):147–156, 2013.

[Ske18] SketchUp. Sketchup. `https://www.sketchup.com`, 2018. Accessed: 2018-09-01.

[SMPZ15] Roy Shilkrot, Pattie Maes, Joseph A. Paradiso, and Amit Zoran. Augmented airbrush for computer aided painting (CAP). *ACM Trans. Graph.*, 34(2), 2015.

[SP06] Raphael Straub and Hartmut Prautzsch. Creating optimized cut-out sheets for paper models from meshes. 2006.

[SP11] Raphael Straub and Hartmut Prautzsch. Creating optimized cut-out sheets for paper models from meshes. Technical report, Karlsruhe Institute of Technology, 2011.

[SP13] Yuliy Schwartzburg and Mark Pauly. Fabrication-aware design with intersecting planar pieces. *Comput. Graph. Forum*, 32(2):317–326, 2013.

[SPR06] Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, 2006.

[SPSH+17] Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z. Kovalsky, and Yaron Lipman. Geometric optimization via composite majorization. *ACM Trans. Graph.*, 36(4):38:1–38:11, July 2017.

[SRML09] Xianfang Sun, Paul L. Rosin, Ralph R. Martin, and Frank C. Langbein. Bas-relief generation using adaptive histogram equalization. *IEEE Trans. Vis. Comput. Graph.*, 15(4):642–653, 2009.

[SS15] Jason Smith and Scott Schaefer. Bijective parameterization with free boundaries. *ACM Trans. Graph.*, 34(4):70:1–70:9, 2015.

[STBG12] Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel, and Markus H. Gross. Computational design of rubber balloons. *Comput. Graph. Forum*, 31(2):835–844, 2012.

[STC+13] Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus Gross. Computational design of actuated deformable characters. *ACM Trans. Graph.*, 32(4):82:1–82:10, 2013.

[STK+14] Mélina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun, and Markus H. Gross. Designing inflatable structures. *ACM Trans. Graph.*, 33(4), 2014.

[STL06] Idan Shatz, Ayellet Tal, and George Leifman. Paper craft models from meshes. *The Visual Computer*, 22(9-11):825–834, 2006.

[SVWG12] Justin Solomon, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. Flexible developable surfaces. *Comput. Graph. Forum*, 31(5):1567–1576, 2012.

[SWH07] Olaf Schenk, Andreas Wächter, and Michael Hagemann. Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Computational Optimization and Applications*, 36(2-3):321–341, 2007.

[Tac10] Tomohiro Tachi. Origamizing polyhedral surfaces. *IEEE Trans. Vis. Comput. Graph.*, 16(2):298–311, 2010.

[Tar12] Marco Tarini. Cylindrical and toroidal parameterizations without vertex seams. In *J. Graphics Tools* [GSA12], pages 144–150.

[TBTB12] Jean-Marc Thiery, Bert Buchholz, Julien Tierny, and Tamy Boubekeur. Analytic curve skeletons for 3d surface modeling and processing. *Comput. Graph. Forum*, 31(7-2):2223–2232, 2012.

[TBWP16] Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. Interactive design of developable surfaces. *ACM Trans. Graph.*, 35(2):12:1–12:12, 2016.

[TCS03] Marco Tarini, Paolo Cignoni, and Roberto Scopigno. Visibility based methods and assessment for detail-recovery. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 60. IEEE Computer Society, 2003.

[TFM+14] Michael T Tolley, Samuel M Felton, Shuhei Miyashita, Daniel Aukes, Daniela Rus, and Robert J Wood. Self-folding origami: shape memory composites activated by uniform heating. *Smart Materials and Structures*, 23(9):094006, 2014.

[The16] Quadraxis Thermo3D. Quadraxis Thermo3D. `http://quadraxis.com/site/?page_id=45`, 2016. Accessed: 2016-01-18.

[TISM16a] Masahito Takezawa, Takuma Imai, Kentaro Shida, and Takashi Maekawa. Fabrication of freeform objects by principal strips. *ACM Trans. Graph.*, 35(6):225:1–225:12, 2016.

[TISM16b] Masahito Takezawa, Takuma Imai, Kentaro Shida, and Takashi Maekawa. Fabrication of freeform objects by principal strips. *ACM Trans. Graph.*, 35(6):225:1–225:12, November 2016.

[tKPSH13] **Christian Schüller**, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. Locally injective mappings. *Computer Graphics Forum (proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing)*, 32(5):125–135, 2013.

[tPG+16] **Christian Schüller**, Daniele Panozzo, Anselm Grundhöfer, Henning Zimmer, Evgeni Sorkine, and Olga Sorkine-Hornung. Computational thermoforming. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 35(4), 2016.

[tPSH14] **Christian Schüller**, Daniele Panozzo, and Olga Sorkine-Hornung. Appearance-mimicking surfaces. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, 33(6):216:1–216:10, 2014.

[tPSH18] **Christian Schüller**, Roi Poranne, and Olga Sorkine-Hornung. Shape representation by zippables. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 37(4), 2018.

[TQZ11] Ping Tan, Long Quan, and Todd Zickler. The geometry of reflectance symmetries. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(12):2506–2520, 2011.

[tra15] transferpaper. Themagictouch cpm 6.2 - hard surface transfer paper. `http://www.themagictouch.com/cpm.html`, 2015. Accessed: 2016-13-04.

[TSP13] Romain Testuz, Yuliy Schwartzburg, and Mark Pauly. Automatic generation of constructable brick sculptures. In Otaduy and Sorkine [OS13], pages 81–84.

[TSS+04] Geetika Tewari, John Snyder, Pedro V. Sander, Steven J. Gortler, and Hugues Hoppe. Signal-specialized parameterization for piecewise linear reconstruction. In *Proc. Symposium on Geometry Processing*, pages 55–64, 2004.

[tur15] Turbosquid. `http://www.turbosquid.com/index.cfm`, 2015. Accessed: 2015-03-01.

[TWS+11] Shigeo Takahashi, Hsiang-Yun Wu, Seow Hui Saw, Chun-Cheng Lin, and Hsu-Chun Yen. Optimized topological surgery for unfolding 3D meshes. *Comput. Graph. Forum*, 30(7):2077–2086, 2011.

[TWZ+17] Ye Tao, Guanyun Wang, Caowei Zhang, Nannan Lu, Xiaolian Zhang, Cheng Yao, and Fangtian Ying. Weavemesh: A low-fidelity and low-cost prototyping approach for 3d models created by flexible assembly. In Mark et al. [MFL+17a], pages 509–518.

[VWRKM13] Kiril Vidimče, Szu-Po Wang, Jonathan Ragan-Kelley, and Wojciech Matusik. OpenFab: A programmable pipeline for multi-material fabrication. *ACM Trans. Graph.*, 32(4), 2013.

[Wan10] Charlie C. L. Wang. From designing products to fabricating them from planar materials. *IEEE Computer Graphics and Applications*, 30(6):74–85, 2010.

[WB97] Andrew Witkin and David Baraff. Physically based modeling: Principles and practice, siggraph course notes, 1997.

[WCKZ12] Meili Wang, Jian Chang, Jens Kerber, and Jian J. Zhang. A framework for digital sunken relief generation based on 3d geometric models. *The Visual Computer*, 28(11):1127–1137, 2012.

[WDB+07] Tim Weyrich, Jia Deng, Connelly Barnes, Szymon Rusinkiewicz, and Adam Finkelstein. Digital bas-relief from 3D scenes. *ACM Trans. Graph.*, 26(3):32, 2007.

[WF88] Gunter Weiss and Peter Furtner. Computer-aided treatment of developable surfaces. *Computers & Graphics*, 12(1):39–51, 1988.

[WGF+18] Kui Wu, Xifeng Gao, Zachary Ferguson, Daniele Panozzo, and Cem Yuksel. Stitch meshing. *ACM Trans. Graph.*, 37(4):130:1–130:14, 2018.

[WHAG15] Christian Weichel, John Hardy, Jason Alexander, and Hans Gellersen. Reform: Integrating physical and digital design through bidirectional fabrication. In Latulipe et al. [LHG15], pages 93–102.

[WK17] Jennifer Weiler and Stacey Kuznetsov. Crafting colorful objects: a DIY method for adding surface detail to 3d prints. In Mark et al. [MFL+17b], pages 2217–2223.

[WPGS18] Katja Wolff, Roi Poranne, Oliver Glauser, and Olga Sorkine-Hornung. Packable springs. *Comput. Graph. Forum*, 37(2):251–262, 2018.

[WT10] Charlie C. L. Wang and Kai Tang. Pattern computation for compression garment by a physical/geometric approach. *Computer-Aided Design*, 42(2):78–86, 2010.

[WW12] Martin Waßmann and Karsten Weicker. Maximum flow networks for stability analysis of lego®structures. In Epstein and Ferragina [EF12], pages 813–824.

[WWB+14] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. Embree: A kernel framework for efficient CPU ray tracing. *ACM Trans. Graph.*, 33(4):143:1–143:8, 2014.

[XLF+11] Shi-Qing Xin, Chi-Fu Lai, Chi-Wing Fu, Tien-Tsin Wong, Ying He, and Daniel Cohen-Or. Making burr puzzles from 3d models. *ACM Trans. Graph.*, 30(4):97:1–97:8, 2011.

[XR16] X-Rite. X-rite. `http://xritephoto.com/`, 2016. Accessed: 2016-01-18.

[Yee00] Yang Li Hector Yee. Spatiotemporal sensistivity and visual attention for efficient rendering of dynamic environments. Master's thesis, Cornell University, 2000.

[YK16] Junichi Yamaoka and Yasuaki Kakehi. Drawforming: An interactive fabrication method for vacuum forming. In Bakker et al. [BHU+16], pages 615–620.

[YK17] Junichi Yamaoka and Yasuaki Kakehi. Protomold: An interactive vacuum forming system for rapid prototyping. In Mark et al. [MFL+17a], pages 2106–2115.

[ZGH+16] Haisen Zhao, Fanglin Gu, Qi-Xing Huang, Jorge A. Garcia Galicia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Connected Fermat spirals for layered fabrication. *ACM Trans. Graph.*, 35(4):100:1–100:10, 2016.

[ZGPR16] Yunbo Zhang, Wei Gao, Luis Paredes, and Karthik Ramani. Cardboardizer: Creatively customize, articulate and fold 3d mesh models. In Kaye et al. [KDL+16], pages 897–907.

[zip17] zipit. zipit store online. `http://www.zipitstore.com/`, 2017. Accessed: 2017-02-01.

[ZK14] Henrik Zimmer and Leif Kobbelt. Zometool rationalization of freeform surfaces. *IEEE Trans. Vis. Comput. Graph.*, 20(10):1461–1473, 2014.

[ZLAK14] Henrik Zimmer, Florent Lafarge, Pierre Alliez, and Leif Kobbelt. Zometool shape approximation. *Graphical Models*, 76(5):390–401, 2014.

[ZLCY12] Long Zeng, Yong-Jin Liu, Ming Chen, and Matthew Ming-Fai Yuen. Least squares quasi-developable mesh approximation. *Computer Aided Geometric Design*, 29(7):565–578, 2012.

[ZP12] Amit Zoran and Joseph A. Paradiso. The freed: a handheld digital milling device for craft and fabrication. In Miller et al. [MBL12], pages 3–4.

[ZTS09] Rony Zatzarinni, Ayellet Tal, and Ariel Shamir. Relief analysis and extraction. *ACM Trans. Graph.*, 28(5), 2009.

[ZTZ17] Yizhong Zhang, Yiying Tong, and Kun Zhou. Coloring 3d printed surfaces by thermoforming. *IEEE Trans. Vis. Comput. Graph.*, 23(8):1924–1935, 2017.

[ZYH+15] Yang Zhou, Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong, and Daniel Cohen-Or. Generalized cylinder decomposition. *ACM Trans. Graph.*, 34(6):Article 171, 2015.

[ZYZZ15] Yizhong Zhang, Chunji Yin, Changxi Zheng, and Kun Zhou. Computational hydrographic printing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2015)*, 34(4), August 2015.

[ZZZY13] Yu-Wei Zhang, Yi-Qi Zhou, Xiao-Feng Zhao, and Gang Yu. Real-time bas-relief generation from a 3D mesh. *Graphical Models*, 75(1):2–9, 2013.