Fabrication-aware 3D Geometry Optimization
(実際のファブリケーション過程を考慮した三次元形状最適化)

by

Kazutaka Nakashima
中島一崇

A Doctor Thesis
博士論文(要約)

Submitted to
the Graduate School of the University of Tokyo
on December 6, 2019
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Information Science and Technology
in Computer Science

Thesis Supervisor: Takeo Igarashi　五十嵐健夫
Professor of Computer Science

ABSTRACT

Digital fabrication is widely spread and used for actual manufacturing in recent years. There are many researches about fabrication not only in the field of engineering, but also computer graphics. However, the applications of the digital fabrication are still limited and it cannot be said that digital fabrication is commonly used.

One of the critical reasons that prevents digital fabrication from becoming common is its inevitable *manual effort*. Digital fabrication procedure consists of following four steps; 1) 3D shape is designed by using CAD software or sculpting software. 2) Pre-processing for actual fabrication is applied to the shape. 3) The shape is fabricated with some fabrication techniques. 4) Post-processing for finishing is applied to the fabricated objects. The 2) pre- and 4) post-processing are done by artists or 3D printing engineers manually, and which takes many efforts. These pre- and post-processing significantly depend on fabrication techniques users employ, and thus the manual efforts are largely varied. To our best knowledge, minimizing such manual efforts are not well studied yet.

In this thesis, we focus on two commonly used fabrication techniques, molding and powder-type 3D printing, and propose practical methods to minimize their pre- and post-processing manual efforts. Specifically, we propose two methods; (i) an interactive method for decomposing the input shape into moldable parts and semi-automatically creating mold piece geometries. (ii) Automatic drain hole position optimization for powder-type 3D printing.

Molding process consists of three following steps; 1) Assemble mold pieces. 2) Pour liquid material such as resin into the void between mold pieces, and solidify the material. 3) Disassemble mold pieces and remove fabricated object from the mold pieces. When we fabricate objects with molding, shapes to be fabricated must be moldable (i.e. can be removed from mold pieces after fabrication). This moldability constraints can be formulated that there is no 3D geometrical feature called *undercut*. However, it is not practical to naïvely judge whether there is any undercut due to its computational costs. Furthermore, the cutting seams are significant disturbances in the visual quality of the assembled models, and thus it is necessary for users to freely add some constraints of the cutting seam positions. To solve the problem, we propose a semi-automatic method to decompose a 3D mesh and design mold pieces with user-defined constraints in interactive speed by approximating the input mesh.

For 3D printing, it is common to hollow the target shape for reducing the amount of material and the cost for materials. Different from fused deposition modeling, which requires infill structure to support the printed object itself, almost all target shape is hollowed when powder-type 3D printing technique is employed. However, with powder-type 3D printing, unsolidified powder material is enclosed by solidified shell, and thus drain holes are needed to recover and reuse such trapped powder. In current practice, the drain holes are manually placed by designers or 3D printing engineers. For specifying drain hole positions, we propose an automatic optimization method that computes drainability on the surface of the input mesh, and finds the best position for drain holes. Computing the drainability with physics-based simulation for powder material movement is not practical due to its expensive computational costs. The proposed method extends the concept of the radiosity method, which is used to compute global illumination efficiently, and approximates the powder movement with matrix operations.

In this thesis, we propose two methods for minimizing the *manual efforts* for typical fabrication techniques. We are confident that our insights accelerate the researches about the *manual efforts* for fabrication.

論文要旨

デジタルファブリケーション技術は、実際の製造業での利活用が開始されるなど、非常に注目を集めている分野である。また、工学の分野に限らずコンピュータグラフィックス (CG) 分野においてもデジタルファブリケーションに関する研究が活発に行われている。このように世間の注目を集め、研究も盛んに行われる分野である一方、デジタルファブリケーションが一般に普及したとは依然として言い難い。

デジタルファブリケーションの普及を妨げる大きな理由の一つとして、『**手間**』が挙げられる。デジタルファブリケーションの過程は、1) CAD ソフトウェアやスカルプトソフトウェアを基に三次元形状を制作する、2) 三次元形状をファブリケーションするための前処理を行う、 3) 3D プリント等を用いてファブリケーションを行う、4) 造形物に対する後処理を行う の四つの工程に分けられる。この中でも特に、2)前処理や4)後処理の大半は手作業で行われており、デジタルファブリケーションを行うアーティストや 3D エンジニアの方にとって大きな負担となってしまっている。更に前処理・後処理の具体的な作業内容はファブリケーション技術に大きく依存しているため、手間の種類も大きく異なる (例:熱溶解積層法による 3D プリントの際のサポート構造の追加や、サポート構造の除去や造形物の表面をヤスリで整える処理)。このような前処理や後処理を対象とした研究はあまり行われてこなかった。

そこで本博士論文では、広く利用されているファブリケーション技術である「型を用いた造形技術」と、「粉末 (例:石膏) を利用する造形技術」における前処理・後処理の手間に注目し、それらを軽減するための技術を提案する。具体的には、(i) 型による造形をするための対象形状の対話的な分割及び、型のデザイン手法、(ii) 粉末タイプの 3D プリントにおける、粉末の排出口位置の最適化手法 の二つの手法を提案する。

型による造形の工程は、1) 型を組み合わせる、2) 組み合わせた型の間に液体素材 (例:レジン) を流し込み、固める、3) 複製された造形物を型から取り出す という三段階に分けることができる。しかし、型を用いる場合、造形物を取り出す際に「型に引っかかることなく取り外す事ができる」ことが常に保証されていなければならない。この制約は、アンダーカットと呼ばれる三次元的な特徴が存在しないように形状を分割する問題として考える事ができる。しかし、アンダーカットの有無をナイーブに判定することは非常に計算コストが高く実用的ではない。更に、分割による切れ目 (分割線) は最終的な見栄えに大きな影響を与えるため、分割線の位置に関する制約をユーザが自由に追加できるかどうかが実用上重要となる。そこで第一の研究として、対象形状の近似処理によるインタラクティブな速度の実現と、ユーザによる分割線の制約を反映可能な半自動的な手法を提案する。

3D プリントにおいて、使用する素材の量を減らす (金銭的コストを低減させる) ために対象形状の中身を空洞化する処理が一般的に行われている。形状保持のための特殊な内部構造が必須となる熱溶解積層法と異なり、石膏分などを利用する粉末タイプの 3D プリント技法では空洞化処理が適用されている。その一方、粉末タイプの 3D プリントの場合、凝固されていない粉末が内部空洞に残留してしまう。このような内部に残留した粉末を除去・再利用するためには三次元形状に粉を排出する穴 (排出口) が必要となる。しかし、排出口を配置する位置は、デザイナや 3D プリントエンジニアが手作業で決定しているのが現状である。そこで、第二の研究として、対象形状の各部位における「粉の排出性能」を計算し、最適な排出口の位置を自動計算する手法を提案する。ただし、物理シミュレーション

を用いた粉末の動きの計算は非常に計算コストが高く、あらゆる部位の排出性能を計算するには不向きである。そこで、提案手法では、大域照明の効率的な計算を目的としたラジオシティ法を応用し、粉末の移動情報を行列形式で近似する手法を提案した。

本博士論文では、デジタルファブリケーションにおける二つの『**手間**』を軽減する手法を提案した。本研究で得られた知見は、いまだ黎明期である『**手間**』に着目した形状の最適化という研究分野の今後のさらなる発展の礎になると信じている。

Acknowledgements

First, I very appreciate my supervisor Takeo Igarashi. He taught me the way to be a good researcher. Specifically, we strongly encouraged me to apply conpetitive research budges, which enables me to do research about the things that I am really interested. Additionally, he gave me very practical advices about attitude to do research, paper writing, and the way to appeal the research contribution to the community. Doing my graduate study under his supervision makes me confident that I would be able to do research in any other research group. I also appreciate my thesis committee members: Reiji Suda, Shinpei Kato, imari Sato, Yoshihiro Kawahara, and Hiromasa Suzuki.

I thank lab members I met during my graduate study. Tsukasa Fukusato, he is a professional researcher in the field of computer graphics, especially in the field of geometry processing. I often discuss with him about the algorithm in depth. In addition, he also helped me a lot with daily life in the university. Daisuke Sakamoto, he is a professional researcher in the field of human computer interaction. He taught me the way to evalueate the interactive system and how to measure human-related factors. I also thank other lab members; Seung-Tak Noh, Hironori Yoshida, Hirofumi Seo, Valentin Nigolian, Katja Wolff, Ichao Shen, Vladimir Romero, Paulo Silva,, Takahiro Sato, Hirotaka Suetake, Shunnki Chou, Kousuke Nakagawa, Kentaro Asai, Zou Zijun, Tomoki Hori, Shugo Miyata, Sai Vinyl Chintalapudi, Ryuto Takano, Toby Chong Long Hin, Cai Zheyuan, Lu Yi, Daiki Harada, Yosuke Fujii, Tomoya Kumazaki, Yuka Takahashi, Akira Koga, Ryohei Shibata, Kai Kubota, Yinan Wang, Syugo Seki, Zhongyuan Hu, Che-An Lin, Koki Toda, Yosuke Kaji, Ryo Sasaki, Morihiro Nakamura, Kazuki Fujiwara, Ichiro Hiraide, Toyoki Kondo, Kazuhiko Yamamoto, Kazuyo Mizuno, Yuki Koyama, Naihui Fang, Kai Shidachi, Hiroaki Mikami, Nobuki Yoda, Kentaro Hayashi, Mageri Filali Maltouf, Hikaru Ibayashi, Yosuke Takami, Sou Araya, Ryohei Suzuki, Takahito Hamanaka, Makoto Nakajima, Naoki Sasaki, Ko Mizoguchi, Koumei Fukahori, Kazuki Ueda, Kouki Nagai, Kazunori Matsuo, Yasunga Fujikado, Yang Xi, Maria larsson, Christian Arzate Cruz, and Nobuyuki Umetani. In addition, I appreciate external researchers gave me precious insight for performing my research; Kenshi Takayama, Takashi Ijiri, Ryoichi Ando, Toshiya Hachisuka, Yoshihiro Kawahara. Particularly, I really appreciate Yuki Koyama, my first paper presented in the international conference [49] is written with Yuki Koyama, Takeo Igarashi, Takashi Ijiri, Shin Inada, and Kazuo Nakazawa. This paper was not published without his great help.

I deeply appreciate Bernd Bickel, he hosted my visit to Institute of Science and Technology Austria (IST Austria) three times. During the visit to IST Austria, I really enjoyed the research and other activities in Austria. Bernd is professional in the field of geometry processing, especially in the field of fabrication, and he taught me very precious advices strongly related with the research field of digital fabrication. I also thank people who discussed with me during the stay in IST Austria; Thomas Auzinger, Ruslan Guseinov, Ran Zhang, Christian Hafner,

Emmanuel Iarussi, Eder Miguel, Chris Wojtan, Camille Schreck, Tomas Skrivan, Georg Sperl, Peter Synak, Patrick Blies, Mortan Bojsen-Hansen, Ewa Gajda-Zagórska, David Hahn, Stefan Jaschke, Radhika Prasasd, Kumar Shivam, and Dorin Gugonatu.

I also appreciate Nobuaki Fukui, who is a professional artist and gave me very precious comments about the problems that artists are facing.

I cannot finish my graduate study without support from Global Creative Leader (GCL) program in the University of Tokyo. Not only the financial support, this program encouraged me to go outside of the academia, and which enlarged my insight. In particular, it gave me a chance to go internship in industrial company and this experience told me the clear difference between the academic research. Japan Science and Technology (JST) ACT-I gave me to discuss professional researchers I cannot meet in the daily life in the University of Tokyo. They also gave me financial support and it really helped me to do research that I feel really excited. I appreciate advisors and researchers who had discussion with me; Masataka Goto, Takeo Igarashi, Daisuka Inoue, Seiichi Uchida, Tetsuya Ogata, Yoshihiro Kawahara, Ken-ichi Kawarabayashi, Shigeru Chiba, Miwako Doi, Kumiyo Nakakoji, Takahiro Hara, Yutaka Matsuo, Shin-ichi Minato.

Lastly, I really appreciate my family Mika and Kazunari Nakashima, Atsuko Nishijima, for their continuous help and encouragement.

Contents

1 Introduction **1**
1.1 Thesis overview . . . 3

2 Related Work **4**
2.1 Designing 3D Shape . . . 4
2.2 Pre-processing for Digital Fabrication . . . 7
2.3 Optimizing Fabrication Process . . . 7
2.4 Post-processing for Finishing . . . 9

3 3D Mesh Decomposition for Molding **12**
3.1 Introduction . . . 12
3.2 Related Work . . . 14
3.3 Problem Formulation . . . 17
3.4 Method . . . 19
3.4.1 Overview . . . 19
3.4.2 Pre-computation . . . 21
3.4.3 Coarse Shell Decomposition . . . 23
3.4.4 Parting Line Computation . . . 24
3.4.5 Part Boundary Smoothing . . . 26
3.5 User Interaction . . . 27
3.6 Mold Generation . . . 29
3.6.1 Detail Restoration . . . 29
3.6.2 Moldability Enforcement . . . 29
3.6.3 Sprues, Runners, and Design Finishing . . . 31
3.7 Results . . . 31
3.8 Limitations . . . 41
3.9 Summary . . . 41

4 Drain Hole Placement Optimization for Powder Recovery **42**

5 Conclusion **43**
5.1 Summary of Our Contributions . . . 43
5.2 Limitations . . . 43
5.3 Future Directions . . . 44

References **45**

Chapter 1

Introduction

In recent years, digital fabrication is widely used along with the spread of 3D printing technologies and 3D printers. The procedure of digital fabrication is roughly separated into four steps (Fig. 1.1); 1) Designing 3D shape by using CAD software or sculpting software. 2) Pre-processing for actual fabrication. 3) Fabricating the shape with some fabrication technologies such as 3D printing and molding. 4) Post-processing for finishing. 1) is the essential step and 3) is the inevitable step in the digital fabrication. Certainly, 2) and 4) are also inevitable, but the amount of time and effort for them could be reduced through sufficient and appropriate support with computer graphics algorithms. Although there exists very wide variety of fabrication technologies, to our best knowledge, there are no general solution for pre- and post-processing reduction. Therefore, we propose novel methods for reducing the amount of time and effort for 2) and 4) for individual fabrication technologies. Specifically, we focus on the pre-processing with molding, and the pre- and post-processing with powder-type 3D printing.

The molding procedure consists of three following steps; 1) Assemble mold pieces. 2) Pour liquid material such as resin into the void enclosed by mold pieces, and the material gets solidified. 3) Disassemble mold pieces and remove fabricated object from the mold pieces. In our scenario, we only consider two-piece molding, which is the commonest and the most efficient kind of molding. For fabricating objects with molding, target shapes must be moldable (i.e. the shapes can be removed from mold pieces after fabrication). Moldable shapes do not have any 3D geometrical features called undercut, which consists of two kinds of 3D geometrical features, overlap and overhang. The overlap is calculated by projecting the target shape along with the direction that the mold piece moves during assembly and disassembly, and this direction is called *parting direction*. The overhang is calculated by checking the sign of the inner product between parting direction and the normal direction on the shape. Except for very simple shapes, almost every shape has undercut and cannot be fabricated by molding. In practice, complex shapes are decomposed into simple moldable parts for dealing with undercuts, and which is basically done manually and takes much effort. However, typical 3D sculpting software are not fully equipped with functionalities for dealing with undercut. For example, the latest ZBrush (ZBrush 2020), which is very popular 3D sculpting software, can detect undercut, but the decomposition itself needs to be done by users manually. Thus, decomposing the shape is still very difficult task especially for novice users, who do not have enough experiences. For this reason, we propose an interactive method for decomposing the input shape into moldable parts and semi-automatic mold piece geometry creation. In this thesis, we first present a surface decomposition algorithm for thin shell replication. Second, we also introduce a solid decomposition algorithm

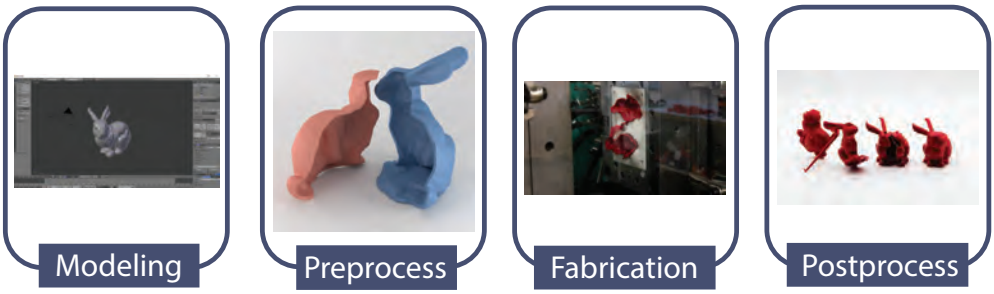


Figure 1.1: Typical step-by-step procedure for digital fabrication. In this figure, we show an example of the procedure with molding.

as a further extension. The surface decomposition algorithm considers the target shape as surface with certain wall thickness. Consequently, the fabricated results are hollowed same as the typical plastic model kits. On the other hand, the solid decomposition algorithm decomposes the target shape into a set of solid (i.e. filled) parts with connectors. These connectors are carefully designed not to interfere the fabrication processes.

This part is removed due to its confidential content.

1.1 Thesis overview

In §2, we review existing studies generally related with digital fabrication. In this chapter, we only review generally related ones and give readers a general trend of the researches about fabrication within the field of computer graphics. In addition to the academic researches, we also introduce many commercial software for designing and optimization for digital fabrication.

In §3, we present the interactive method for decomposing the input shape into moldable parts and semi-automatic mold piece geometry creation. We first introduce the detail of the problem with molding and the decomposition again, and then review existing studies closely related with molding. Then, we present our algorithm in detail. Finally, in this chapter, we present the results obtained by our method, and discuss the limitation and future direction of this study.

§4 is removed dut to its confidential content.

In §5, we again state the potential and difficulty of the digital fabrication, and how our method solves such difficulties. In addition, we also discuss the future extension of our research from more general viewpoint.

Chapter 2

Related Work

As we stated above, digital fabrication consists of four steps; 1) Designing 3D shape. 2) Pre-processing for fabrication. 3) Fabricating the shape. 4) Post-processing for finishing. In this chapter, we review related work of these steps.

2.1 Designing 3D Shape

Designing functional shapes Giving some functionality to 3D printed object is very popular research topic for fabrication [10, 65, 47, 7, 52, 23, 69, 44, 66, 56, 51].

Make it stand [52] is one of the most outstanding research. optimizes the gravitational properties by using carefully designed internal void and deformation. Due to this hollowing and the deformation, the fabricated object is guaranteed to stand without any support structures (Fig. 2.1). Pteromys [65] is also one of the most outstanding research. This optimizes the stability of gliders by using machine learning technique. By using this method, even novice users can design and fabricate a glider that flies well with very unique shape (Fig. 2.2). Wang et al. [69] also optimizes the internal void and put some weight to control the gravitational properties to float in the water with user specified posture. Printone [66] also optimizes the internal void, but their motivation is to optimize the acoustic functionality and create freeform wind instruments. Schumacher et al. [56] and Panetta et al. [51] use micro structure to achieve very wide range of elasticity (Fig. 2.3).

There are also researches that maximally exploit the accuracy of 3D printers and achieve some functionalities [5, 25]. These functionalities cannot be achieved without the accuracy of 3D printers. Auzinger et al. [5] use nano-scale 3D printer to achieve structure color. (Fig. 2.4) Guseinov et al. [25] uses carefully designed small tiles and pre-stretched latex sheet to achieve doubly curved surface.

Personalized fabrication One strong advantage of the digital fabrication is personalization. There are also several studies that enable personalization [75, 37, 9, 6, 59].

Koyama et al. [37] automatically generates personalized connectors for user-specified objects. Their methods allows users to explore the possible connector geometries by very simple mouse interaction. Zhang et al. [75] automatically transfer pre-defined mechanical templetes to user-specified objects. By using thier method, users can create fuctional objects without being aware of geometrical constraints.

Figure 2.1: Make it stand [52] automatically optimizes gravitational properties of the input shape. This figure is adapted from the Figure 1 of [52].

Figure 2.2: Examples designed with Pteromys [65]. They have very unique shape, but all of them fly well. This figure is adapted from the Figure 14 of [65].

Figure 2.3: Elastic object fabricated with [56]. Color on the leftmost image represents user-specified elasticity parameters. This figure is adapted from the Figure 1 of [56].

Figure 2.4: Printed object with structure color by using [5]. A silhouette of a teapot is colored in blue, and the background is colored in red. Both color are structure color and no color pigment is used. This figure is adapted from the Figure 1 of [5].

Commercial software For complete the reference for design process, I also mention the commercial software for 3D modeling. Roughly speaking, such software could be classified into two categories, computer aided design (CAD) software and 3D sculpting software. The former is mainly used to design artificial, man-made shapes such as gears. For example, SolidWorks, Autodesk Fusion 360, Blender, and OpenSCAD are common CAD softwares. On the other hand, 3D sculpting software is mainly used to create freeform shapes such as figures or statues. For example, ZBrush, Sculptris, and Geomagic Freeform are common 3D sculpting software. Note that this classification is just for reference, and 3D sculpting software could use for designing man-made shapes. In addition, in recent years, several software have functionalities for designing both man-made shapes and freeform shapes. For instance, ZBrush has functionality (Zmodeler) for man-made shapes design and Blender also has sculpt mode for freeform shape.

2.2 Pre-processing for Digital Fabrication

Support structure for additive manufacturing Soon after 3D printing technique, especially fused deposition modeling (FDM), become common, many studies that optimizes support structure are reported [17, 55, 67, 27, 68].

Dumas et al. [17], Schmidt et al. [55], and Vanek et al. [67] proposed efficient support structures with different approaches (Fig. 2.5). On the other hand, He et al. [27] and Vanek et al. [68] decomposes the target shape into multiple parts that can be fabricated without support structure.

Traditional fabrication technique Computer graphics researches for fabrication are not limited to fabricate objects with computer numerical control (CNC) machines. There are researches that pre-process the target shape and fabricate it by using traditional fabrication techniques [26, 2, 43, 48, 1, 63].

Focusing on molding, there are wide variety of research to fabricate single object with multiple mold pieces [26], single object with flexible mold [43], single object with multiple deformable mold pieces [2, 1]. There are also drastic deformation algorithm for molding [63], and which even changes its topology.

Color printing In recent years, 3D printing technologies are still evolving, and they can print color objects, and which invokes the new research topics in the field of fabrication [18, 64]. These researches solve the problem of color bleeding caused by the semi-transparency of the printing material. They cancel color bleeding (i.e. the effect of subsurface scattering) by optimizing the color within the printed objects (Fig. 2.6).

2.3 Optimizing Fabrication Process

Reducing printing time and material usage One straightforward optimization of fabrication process is hollowing inside to reduce the amount of material used [72, 41]. Lu et al. [41] optimize the infill structure (certain pattern filling the hollowed internal void) while keeping the balance between material usage and strength (Fig. 2.7). There are other approaches optimize the fabrication process [45, 46, 8, 33].

WirePrint [45] drastically reduce the printing time and material usage by only printing the wireframe of the target shape (Fig. 2.8). Printed object is not sufficient for actual use, but it is enough to roughly check the design and the

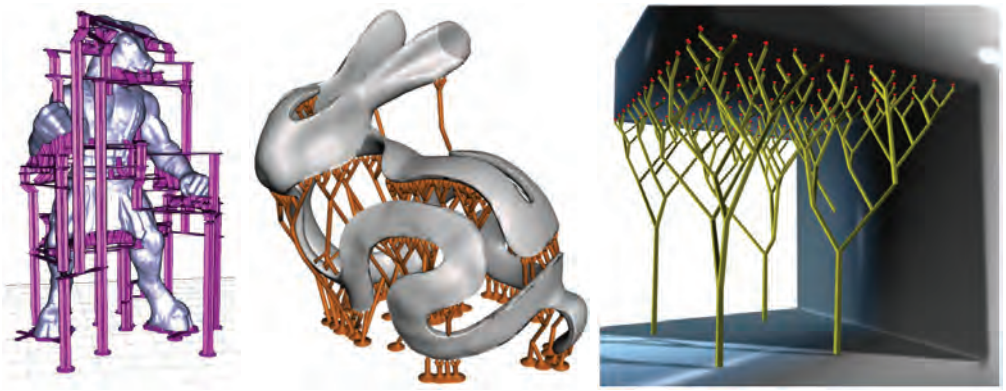


Figure 2.5: (Left) support structure generated by [17]. (Center) support structure generated by [55]. (Right) support structure generated by [67]. This figure is adapted from the Figure 10 of [17], Figure 1 of [55], and Figure 7 of [67].

Figure 2.6: Printed result by using [18]. These results have finer detail than naïve color printing. This figure is adapted from the Figure 1 of [18].

scale, which would speed-up the iteration during the design process. faBrickation [46] and TrussFormer [33] have similar concept that only fabricates small parts, which is combined with other pre-fabricated objects (LEGO blocks for [46] and PET bottles for [33], Fig. 2.9).

Optimize toolpath for printing The quality of the printed objects and the printing time heavily depends on the toolpath for printing (e.g. G code). There are several researches that optimizes the toolpath [71, 19, 16, 4].

Wang et al. [71] and Alexa et al. [4] vary the thickness of the printing layers while keeping the effect on the surface qualilty of the printed object minimum. On the other hand, Etienne et al. [19] allow curved layers in +Z direction. Dai et al. [16] change the posture of the (partially) printed object by robot arms (Fig. 2.10). These methods only optimizes the toolpath, and do not make effect on the input shapes.

2.4 Post-processing for Finishing

The researches about support structure we already mentioned above indirectly optimizes post-processing by relieving the cost for support removal.

Apart from the researches about support structure, there are also other researches that help users to post-process the fabricated objects. For instance, when the shape is printed as a set of parts, users need to assemble the parts for the final complete object. [61, 60, 21] fabricate interlocking objects to relieve the manual effort to assemble after the fabrication.

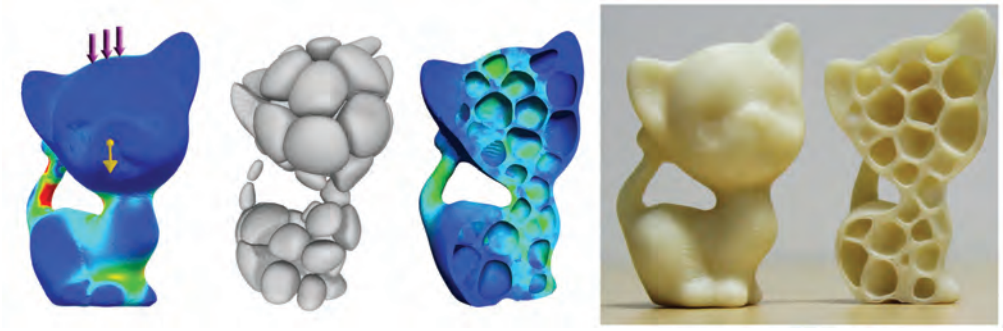


Figure 2.7: Infill structure generated by [41]. Their method save the material while keeping the printed object strong. This figure is adapted from the Figure 1 of [41].

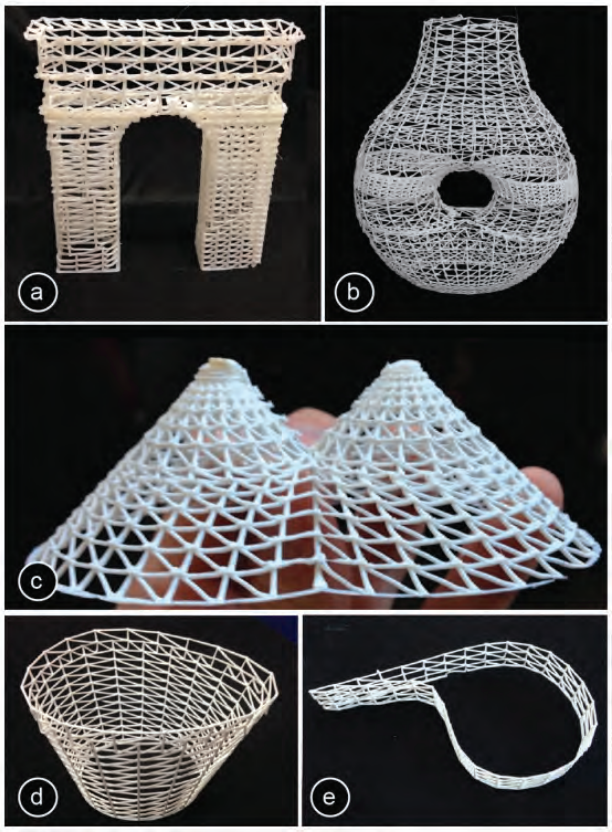


Figure 2.8: Examples printed by using WirePirnt [45]. This figure is adapted from the Figure 11 of [45].

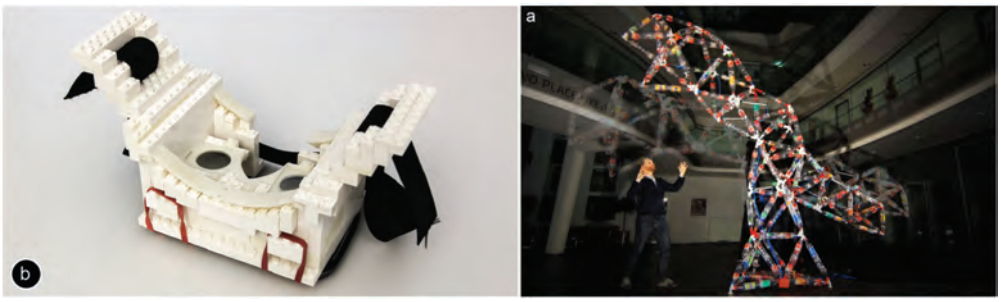


Figure 2.9: Examples by faBrickation [46] and TrussFormer [33]. This figure is adapted from the Figure 8 of [46] and Figure 1 of [33].

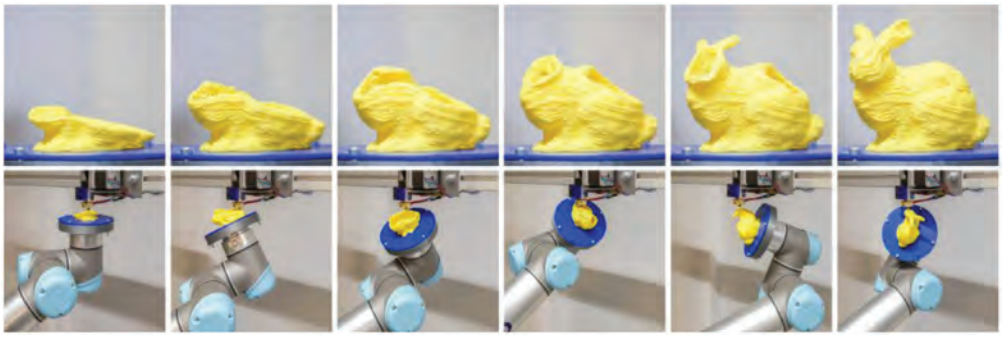


Figure 2.10: Printing procedure by using [16]. The printing platform is mounted on a robot arm and the posture is changed during the printing. This figure is adapted from the Figure 1 of [16].

Chapter 3

3D Mesh Decomposition for Molding

Molding is a popular mass production method, in which the initial expenses for the mold are offset by the low per-unit production cost. However, the physical fabrication constraints of the molding technique commonly restrict the shape of moldable objects. For a complex shape, a decomposition of the object into moldable parts is a common strategy to address these constraints, with plastic model kits being a popular and illustrative example. However, conducting such a decomposition requires considerable expertise, and it depends on the technical aspects of the fabrication technique, as well as aesthetic considerations. We present an interactive technique to create such decompositions for two-piece molding, in which each part of the object is cast between two rigid mold pieces. Given the surface description of an object, we decompose its thin-shell equivalent into moldable parts by first performing a coarse decomposition and then utilizing an active contour model for the boundaries between individual parts. Formulated as an optimization problem, the movement of the contours is guided by an energy reflecting fabrication constraints to ensure the *moldability* of each part. Simultaneously, the user is provided with editing capabilities to enforce aesthetic guidelines. Our interactive interface provides control of the contour positions by allowing, for example, the alignment of part boundaries with object features. Our technique enables a novel workflow, as it empowers novice users to explore the design space, and it generates fabrication-ready two-piece molds that can be used either for casting or industrial injection molding of free-form objects.

3.1 Introduction

In recent years, the rise of 3D printing technology has democratized the fabrication of complex shapes. Nevertheless, classical manufacturing approaches, such as molding, are often the preferred choice because of two main reasons: they scale extremely well with the amount of required copies, and they draw from a wide range of available base materials. Unfortunately, in contrast to 3D printing, which seemingly requires only a few clicks to replicate an object, obtaining a mold is much more challenging.

In this chapter, we focus on designing re-usable rigid molds for *two-piece molding*, the simplest and most cost-efficient variant of molding. In two-piece molding, as illustrated in Fig. 3.2, two rigid mold pieces form a cavity filled with a liquid or pliable material, i.e., resin, plastic, metal, or ceramics, which then solidifies. Subsequently, the mold pieces are separated, and the cast object is released by a linear translation along a so-called *parting direction (PD)*. To make the removal possible, the top and bottom surfaces of the fabricated object must be representable as height fields for precluding overhangs and overlaps. Previous work

Figure 3.1: From left to right: We propose a method to decompose an input shape (surface mesh) into molding-ready shell pieces and to generate the corresponding 3D printable molds. Using, for example, an injection molding machine, we can fabricate many physical replica at low cost and in a short amount of time.

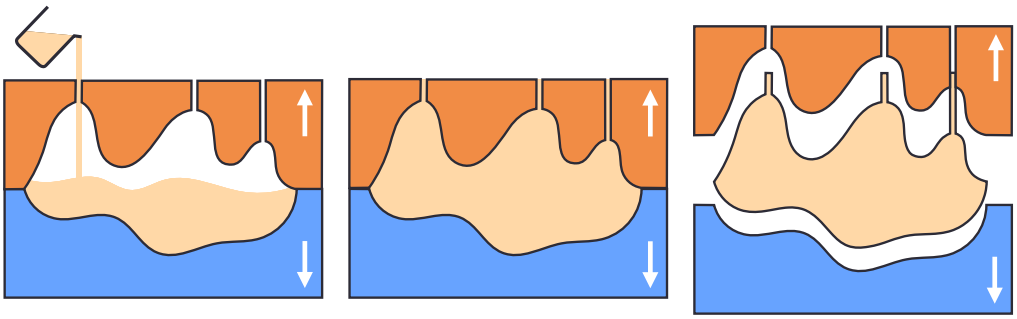


Figure 3.2: Molding procedure with a two-piece mold. First, the mold pieces are assembled to form a cavity, into which the liquid object material is casted (left). Pockets of air are avoided by outlets in the top mold piece. After material solidification (center), the mold pieces are removed, and the object (right) remains.

has extensively investigated finding the optimal parting directions [32], or methods to deform a given shape [26], to obtain a valid height field configuration. Nevertheless, this constraint of molding makes faithfully reproducing many man-made and most organic objects as a single piece impossible. While introducing multi-piece molds and multiple parting directions helps overcome this limitation, our goal is to avoid adding complexity to the molding process and keeping our molds compatible with simple injection molding machines.

A suitable strategy to overcome the aforementioned restrictions imposed by two-piece molds is the decomposition of the desired shape into a set of parts, in which each part itself satisfies the fabrication constraints (see Fig. 3.3). Such a decomposition needs to reconcile the technical guidelines for mold design with aesthetic considerations regarding the placement of boundaries between the individual parts. This task is highly complex, as the designer needs to anticipate the effect of decomposition on the manufacturability, aesthetics, and overall size of the mold, while trying to keep the number of individual parts small. Therefore, in practice, this task is usually reserved for experts.

We propose a computational approach that allows novice users to design a two-piece mold. Our interactive technique supports exploring the design space and finding a decomposition of a free-form shape into a set of solid parts. We assume that a target shape is provided as an orientable surface mesh. The inner surface is automatically generated as a thin-shell offset surface. In contrast to a solid fill, a thin shell reduces the required amount of material and the weight of the final object.

In this chapter, we propose an energy formulation to quantify the quality of a part and solve a constrained optimization problem, in which the moldability is taken into account for each part. On this basis, for the purpose of decreasing the assembly cost, our method attempts to minimize the number of parts. We utilize a dedicated optimization method to compute the shape of the individual parts and consider both aesthetic and fabrication constraints. In addition, the integration of subjective user preferences by providing a set of intuitive interactive tools to edit the decomposition, steer the optimization process, and explore the solution space ensures that the designer stays in control over aesthetic considerations. We demonstrate the viability of our approach on a range of input shapes by creating molds that are used to fabricate a variety of models.

3.2 Related Work

Computer-aided design and molding A comprehensive introduction and overview of recent techniques can be found in the work of Kazmer [31]. Because of its industrial relevance, computer-aided mold design has received significant attention [22]. While a mold design might be influenced by a multitude of aspects, such as cost, fabrication speed, required amount of casting material, material flow during molding, life-span of the mold, and aesthetics, a fundamental task in designing molds is to geometrically verify and model shapes that can be fabricated with this technique. In the remainder of this section, we will focus on this aspect.

Commonly, moldability is tested by searching for parting directions that allow the opening of the mold without interlocking with the fabricated object or the mold itself [32, 13, 74, 28]. Optimal parting directions are usually determined by trying to automatically recognize and avoid undercut features [50]. Additionally, criteria determining the quality of parting lines at which individual mold pieces join have been modeled to analyze given designs [54], or to even compute these parting lines automatically, with a parting direction as input [39].

Table 3.1: Comparison between our method and related work on mold design.

	[26]	**[43]**	**Our method**
Object parts	1	1	multiple
Mold pieces per part	multiple	1	2
Mold type	rigid	flexible	rigid
Mold assembly	complex	complex	simple
Representation	surface-only	surface-only	vol.-aware
Thin-shell supported	no	no	yes
Interactive editing	no	no	yes

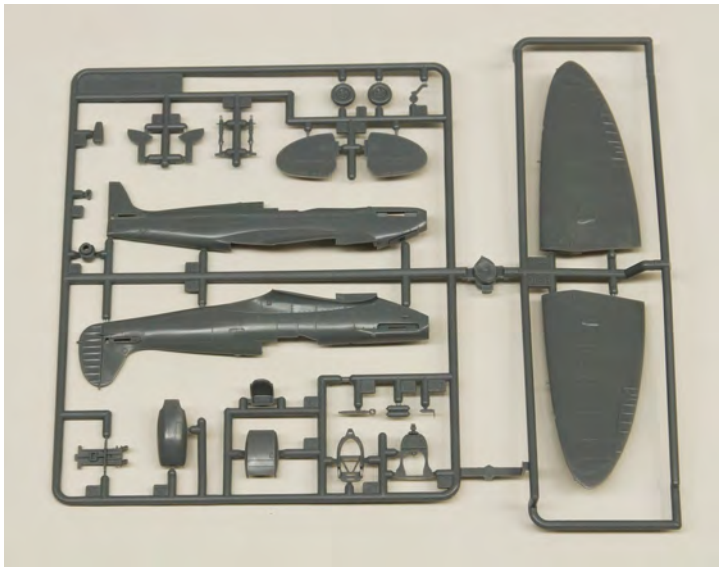


Figure 3.3: Example of model airplane decomposed into a set of parts, fabricable with a two-piece mold.

To extend the range of moldable shapes, some authors have explored the use of multi-piece molds for dealing with concave regions that tend to generate overhangs [53, 40, 26]. These methods decompose an input surface into several height field patches for which corresponding mold pieces are computed. The approach by Lin et al. [40] starts by identifying a small set of parting directions, and then follows a set of rules to assign surface facets to mold-piece regions. Herholz et al. [26] used a graph cut-based approach to decompose and, as necessary, deform the surface of an object to approximate free-form geometry with height fields.

Common to all these approaches is that they fall into the category of *surface segmentation* methods. In general, for these methods, no obvious extension to the segmentation of solids with enclosed voids exist, because, for instance, for an outer surface patch, they have no knowledge of parting surfaces inside the volume and the orientation of inner surfaces, making the identification of the moldability of the resulting parts infeasible. Furthermore, a valid height field surface segmentation cannot be easily converted into moldable shells because of the potential intersections of extruded shells. This case is especially challenging in the presence of thin features, as shown in Fig. 3.6, because inner surfaces might touch and therefore require these regions to be treated as a single volume. To prevent these issues, our proposed method performs a volume-based decomposition of the shape.

Relaxing the rigidity constraint, Malomo *et al.* [43] used a computational approach to design flexible molds that can undergo considerable deformation during removal. However, the molds must be elaborately sealed before casting, thus compromising the time saving potential for which molding is used in the first place. Additionally, the extension to thin-shelled objects is non-trivial. Moreover, flexible molds are not suited for injection molding, as they cannot withstand the high pressure during injection. Tab. 3.1 summarizes the main differences and similarities between our method and the aforementioned approaches.

Decomposition for fabrication Fabricating shapes as sets of parts that are subsequently assembled is a prominent strategy in manufacturing. In the field of computer graphics, previous work includes the decomposition into pyramidal pieces, which can be 3D printed without any support structure because of the lack of overhangs [27]. Wang *et al.* [70] decomposed a polygonal model to minimize 3D printing artifacts by considering anisotropic printing resolution. Furthermore, decomposition allows users to print large shapes beyond the build size limitations of commodity 3D printers [42, 68, 14, 73]. Alemanno *et al.* 2014 [3] proposed a technique to decompose a 3D digital shape into a set of interlocking pieces that can be represented as simple height fields. It relies on a manually provided very coarse polyhedral approximation of the input shape, which determines the decomposition. Currently, the problem of finding such a coarse approximation of a geometric shape is an open problem and requires further investigation to be solved in an automatic and robust manner on complex 3D models. Gao et al. [24] introduced a multi-directional printing system with a revolving cuboidal build platform. While they also proposed a segmentation method into axis-aligned height field blocks to reduce the print and support material use, our problem setting differs significantly. Instead of being bound to an axis, we are trying to obtain a segmentation that ensures moldability, while exploring a large number of parting directions and free-form cuts.

Commercial software for molding Tools for computationally designing molds have also been commercially explored because of their significant industry relevance. Popular products, such as Moldflow, Moldex3D, or CADMOULD, simulate the molding process and are used for computer-aided engineering (CAE) verification and design improvement. However, these workflows are usually intended for experts, require proficient knowledge of the CAD packages they are interfacing with, such as Autodesk Fusion 360, SolidWorks, or CATIA, and start with an initial mold design created by an engineer. By contrast, our tool can fully and automatically decompose a given free-form shape into shell parts and provide a free-form parting line per part for efficient two-piece molding. Its interface is simple enough to be used without extensive training. Our approach is complementary and, in theory, could be combined with existing CAE tools.

3.3 Problem Formulation

As input geometry, we assume a closed surface $\mathcal{S}_{\text{input}}$, which we want to fabricate using two-piece molding. We require the surface to be a discrete, connected, and suitably oriented manifold without self- or pairwise intersections and given as a triangle mesh. As output, we provide a set $\mathcal{P} = \{P_i : i = 1, \dots, N_\mathcal{P}\}$ of $N_\mathcal{P}$ *parts*, which constitute a decomposition of a thin shell volume $\widetilde{V}$ of the potentially deformed input shape. The decomposition is both disjoint and covers all of the shell $\widetilde{V}$, i.e.,

$$P_i \subset \widetilde{V}, \quad P_i \neq \emptyset, \quad \bigcup_{i=1}^{N_\mathcal{P}} P_i = \widetilde{V}, \quad P_i \cap P_j = \emptyset \quad \text{for } i \neq j.$$

Each part P_i fulfills the fabrication constraints of two-piece molding and its surface is divided into two patches M_i^+ and M_i^-. The two patches correspond to the surface of the two mold pieces and they share the identical boundary (parting line), and again they are connected (but not necessarily simply connected), oriented, and free of intersections.

Additionally, each part is equipped with its parting direction $\boldsymbol{d}_i$ along which the mold pieces are assembled and disassembled. Note that we do not allow different parting directions for the two mold pieces to reduce fabrication complexity.

A fundamental constraint of molding is the required absence of overhangs and overlaps, which can inhibit the removal of the mold after the liquid material has been cast and solidified. Thus, the faces M_i^+ and M_i^- of the two mold pieces of a part P_i with parting direction $\boldsymbol{d}_i$ need to constitute height fields. As a necessary condition, we require for all triangles $t^+ \in M_i^+$ (resp. $t^- \in M_i^-$) that their outwards pointing normals $\boldsymbol{n}_{t^+}$ (resp. $\boldsymbol{n}_{t^-}$) satisfy

$$\boldsymbol{d}_i \cdot \boldsymbol{n}_{t^+} \geq 0 \quad (\text{resp. } \boldsymbol{d}_i \cdot \boldsymbol{n}_{t^-} \leq 0)$$

as illustrated in Fig. 3.5 (left, red). In practice, a completely vertical wall is inappropriate for molding (Fig. 3.5 (middle)), and all triangles must be tilted with a small angle ϕ, which is known as the *draft angle* (Fig. 3.5 (right)).

$$\arccos(\boldsymbol{d}_i \cdot \boldsymbol{n}_{t^+}) \leq \frac{\pi}{2} - \phi \quad (\text{resp. } \arccos(\boldsymbol{d}_i \cdot \boldsymbol{n}_{t^-}) \geq \frac{\pi}{2} + \phi) \tag{3.1}$$

Global overlaps can occur (see Fig. 3.5 (left, blue)), and they are characterized by an overlap of the triangles of either M_i^+ or M_i^- when projected via $p_{\boldsymbol{d}_i}(x) =$

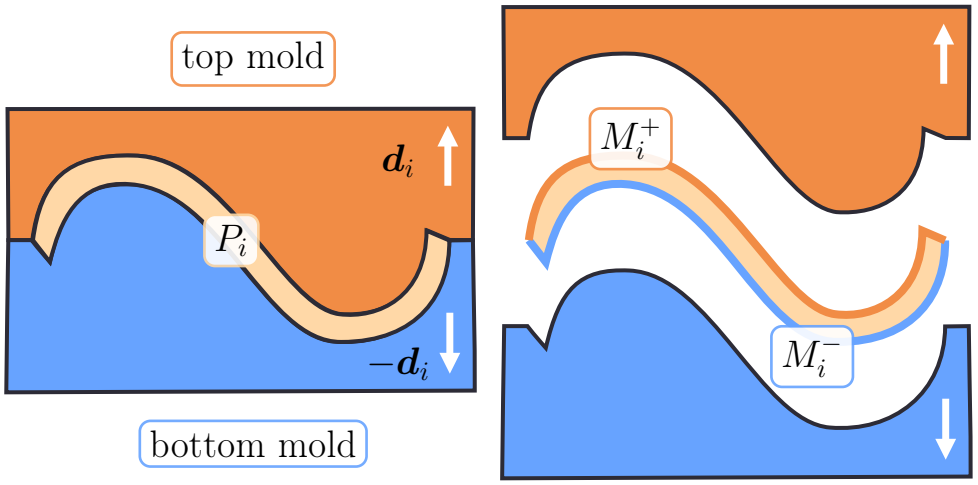


Figure 3.4: Definition of notations used in this chapter. We use P_i for each part, $\boldsymbol{d}_i$ and $-\boldsymbol{d}_i$ for parting direction for top mold and bottom mold, respectively. Also, we use M_i^+ for surface of P_i that touches with the top mold. Similarly, we use M_i^- for surface that touches with the bottom mold.

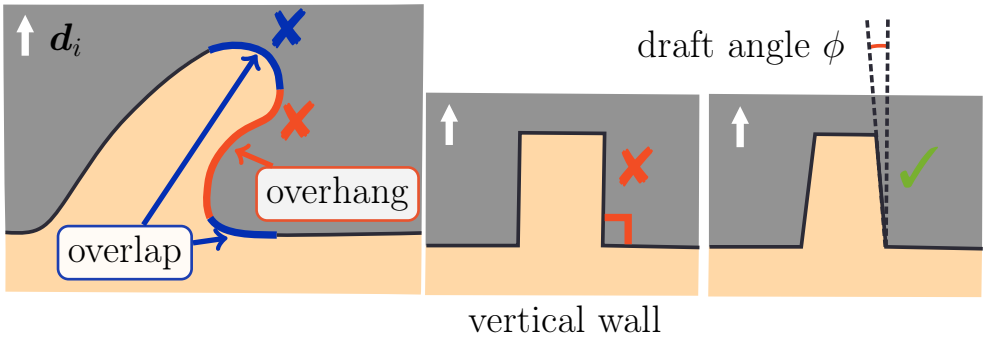


Figure 3.5: Restrictions of molding. Overhang constraint (left, red): all normals of a mold face have to point into the same hemisphere as the parting direction $\boldsymbol{d}$. Global overlap constraint (left, blue): self-intersections when projecting the face along its parting direction $\boldsymbol{d}$ must not occur. Draft angle constraint (right): all triangles must be slightly tilted so that angles between parting direction $\boldsymbol{d}$ and their normal is smaller than $\pi/2$.

$(I - \boldsymbol{d}_i \boldsymbol{d}_i^{\mathrm{T}})\ x$ along the corresponding parting direction $\boldsymbol{d}_i$. To avoid this, we require that

$$p_{\boldsymbol{d}_i}(t_j) \cap p_{\boldsymbol{d}_i}(t_k) = \emptyset \tag{3.2}$$

for all pairs of disjoint triangles t_j and t_k of either M_i^+ or M_i^-. We call this kind of violation (i.e., overhang and overlap) *undercut*.

3.4 Method

The design of our method for solving the decomposition problem stated in the previous section is based on the requirement of providing an *interactive* user experience. Interactivity is essential for exploring the design space, understanding the trade-offs between aesthetic considerations and fabrication limitations, and for achieving a subjectively satisfying result while still assuring moldability. Because of the computational complexity of the problem, obtaining feasible solutions in short time frames is generally not possible. We use two key strategies to mitigate this problem: (i) we perform the decomposition on a lower resolution and then restore the fine details from the original input surface $\mathcal{S}_{\text{input}}$ as a post-process (see §3.4.2 and §3.6) and (ii) we follow a coarse-to-fine approach. First, we perform a coarse decomposition by using region growing based on triangles as atomic elements. Second, we optimize and smooth the boundary by using an active contour representation that can move continuously over the surface to achieve a visually preferable and easier moldable decomposition.

In contrast to automatically trying to find a global optimum from scratch, such as, for example, using graph cut-based approaches [26] in which small user input changes could lead to a drastically different solution, we advocate for an approach that has three key advantages that make it suitable for user interaction: (i) If desired, users can directly control the number of parts and their boundaries, which enables them to reflect their intent on the obtained result; (ii) user interaction affects the decomposition locally, and this makes it possible to concentrate on editing the region of interest; and (iii) the decomposition procedure and the effects of user interactions are visualized at interactive rates, thereby providing intuitive editing operations to non-expert users.

3.4.1 Overview

The workflow of our method is illustrated in Fig. 3.6 and Alg. 1. We begin with remeshing the input mesh $\mathcal{S}_{\text{input}}$, generating a low-resolution representation $\mathcal{S}_{\text{low}}$. Then, we extrude the surface $\mathcal{S}_{\text{low}}$ and generate a volumetric representation V to be decomposed (§3.4.2). During the shell generation, thin features are detected and marked to be fabricated as solids because they might be too fragile when split in separate shells. (Fig. 3.6 (middle left)). Over this shelled object, we perform a coarse decomposition (§3.4.3) by using a volume-aware region growing strategy, and generate a set of *parts* (Fig. 3.6 (middle)). For generating a decomposition with a small part count $N_{\mathcal{P}}$, we first try to decompose the object into two parts, and then increase the part count one by one until the moldability energy becomes smaller than the user-defined threshold γ. For each part, we compute the parting line for two-piece molding using a graph cut-based surface classification (§3.4.4). This results in a pair of connected surface *patches* that define the top and bottom surface of the part in the mold (orange and blue in Fig. 3.6 (middle right)). Using the obtained solution as an initial guess, we then represent the boundaries between parts as active contours that can continuously slide over the surface. We optimize

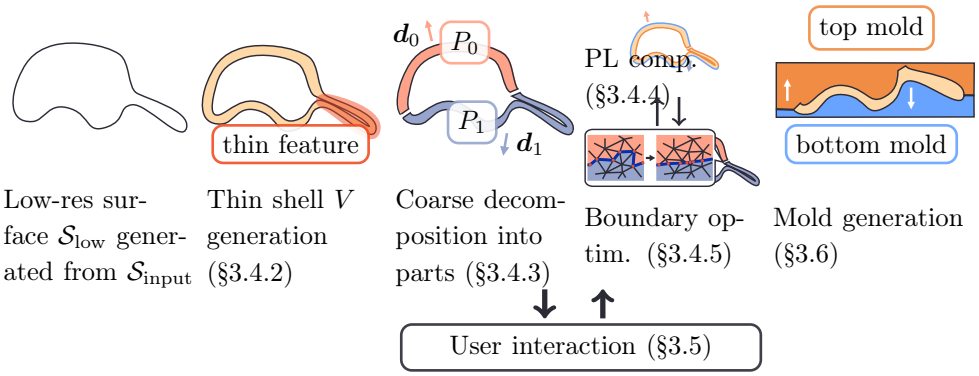


Figure 3.6: We first compute an offset surface to obtain a thin-shell object (left). During the shell generation, we detect thin features (red) to be fabricated as solids (center left). Then, the shell object is decomposed into parts, and the parting line and part boudaries are optimized (center, center right). Users can reflect their design goal to the decomposition by user interaction. Finally, mold geometries are generated from the parting line (right).

these boundaries based on an energy that quantifies their smoothness and the moldability of the parts (§3.4.5). During this optimization, the parting lines of individual parts are updated as well.

Both before and after the coarse part decomposition and subsequent boundary optimization, users can add design constraints to reflect their intent (§3.5), while the solution is automatically updated. Finally, our system performs post-processing for restoring detail of $\mathcal{S}_{\text{input}}$ (§3.6.1) and moldability verification (§3.6.2), and then generates the mold geometry for fabrication. Runner structures (pipes for liquid material flow) are automatically added. After the molds are milled or 3D printed, a large number of copies can be efficiently fabricated using either resin casting or injection molding (§3.6.3).

3.4.2 Pre-computation

In the pre-computation stage, we prepare a volumetric shape representation, which is then used for decomposing the shape into moldable parts. To ensure the interactivity of our method, we start by computing a low-res representation $\mathcal{S}_{\text{low}} = (V_{\text{low}}, T_{\text{low}})$ of our input surface $\mathcal{S}_{\text{input}}$ with the use of instant field-aligned meshes [30], where V_{low} and T_{low} denote vertices and triangles of $\mathcal{S}_{\text{low}}$, respectively. In our experiments, a triangle count of $N_\Delta \lesssim 10000$ provided a good trade-off between speed and accuracy. We also remesh $\mathcal{S}_{\text{input}}$ as necessary to obtain a high-quality, uniformly meshed high-resolution representation.

We aim to fabricate our shape as a thin-shelled object with an approximate user-specified wall thickness ω, which extends into the interior. To generate the internal surface of the thin shell, we push the vertices V_{low} inward along the inverse of the gradient direction of the signed distance field (< 0 inward and > 0 outward). In our implementation, we use the method of Jacobson et al. [29] for signed distance calculation. While this approach might result in an internal surface with potentially degenerated triangles, flipped triangles, or self-intersection, our technique is robust against such deficiencies. Because of our internal surface generation, we have bijective mappings $f_V : V_{\text{low}} \to V_{\text{in}}$ and $f_T : T_{\text{low}} \to T_{\text{in}}$, where V_{in} and T_{in} denote the vertices and triangles of the internal surface, respectively. We define our *atomic elements* as triangle prisms whose top face is a triangle of T_{low}, and whose bottom face is the corresponding triangle of T_{in}. Note that the quadrangle side faces of our atomic elements (i.e., triangle prisms) T_{side} are triangulated to ensure that all polygons are triangles, and we also have a surjective mapping $g : T_{\text{side}} \to T_{\text{low}}$.

Thin features (e.g., ears of bunny) require special attention during offsetting, as shells intersect. While an option would be to restrict the wall thickness to prevent intersections, in practice this might lead to thin and fragile parts. Instead, we fabricate such parts as solids (i.e., without an internal void) to increase strength and to improve the flow of the liquid material during the casting process. To detect thin features, we use a shape diameter function (SDF)-based segmentation [58] over $\mathcal{S}_{\text{low}}$. For a point on the surface of the shape, its SDF value is twice the approximated distance to the medial axis. For robust detection, we identify segments whose minimal SDF value is smaller than twice the desired wall thickness ω and whose average SDF value is also smaller than three times ω. The label for thin features are propageted from triangles $\in T_{\text{low}}$ to atomic elements.

During the whole decomposition, we use uniformly sampled directions $\mathcal{D}$ as candidates of the parting directions. Thanks to this pre-defined set of candidate directions, we can pre-compute overhangs and overlaps to drastically speed up our workflow.

Algorithm 1 Abstract workflow

Input: watertight and self-intersection free mesh $\mathcal{S}_{\text{input}}$
generate low-res representations $\mathcal{S}_{\text{low}}$ from $\mathcal{S}_{\text{input}}$ (§3.4.2)
generate columetric representation from $\mathcal{S}_{\text{low}}$ (§3.4.2)
$N_{\mathcal{P}} \leftarrow 2$
repeat
  user adds design constraints (§3.5)
  repeat decomposition into $N_{\mathcal{P}}$ parts
    volume-aware coarse decomposition (§3.4.3)
    boundary smoothing (§3.4.5) with parting line comp. (§3.4.4)
    if moldability energy $E_P >$ threshold γ **then**
      $N_{\mathcal{P}} \leftarrow N_{\mathcal{P}} + 1$
    end if
  until moldability energy becomes smaller than threshold γ
until user satisfied with the decomposition
generate mold geometry (§3.6)
fabricate target shape with two mold pieces (§3.6.3)

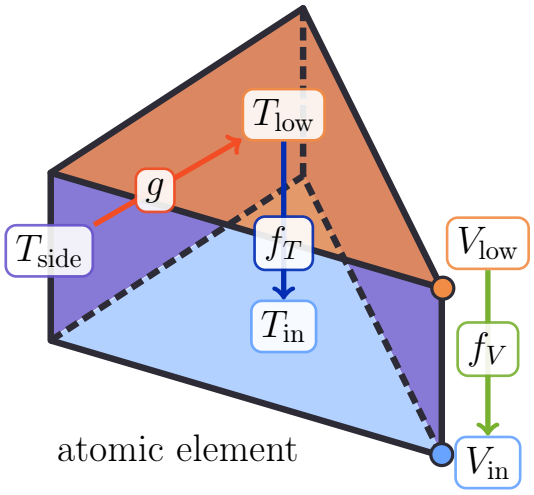


Figure 3.7: Definition of our atomic element for the decomposition. Each atomic element is represented as a triangle prism, and we define three mappings, g, f_T, f_V. g maps a side quadrangle (purple) to corresponding top triangle (orange). f_T maps a top triangle (orange) to corresponding bottom triangle (light blue). f_V maps a vertex adjacent to top triangle to corresponding vertex adjacent to bottom triangle.

3.4.3 Coarse Shell Decomposition

Estimating the minimum required number of parts for a moldable decomposition is a challenging problem. Fekete and Mitchell [20] showed that deciding if a 3D model of genus 0 can be decomposed into k polyhedra that resemble height fields is NP-hard. Therefore, we use a simple yet reasonable greedy approach to decompose the target shape into $N_{\mathcal{P}}$ moldable parts. We start with $N_{\mathcal{P}} = 2$ (recall that our target shape is a shelled object and therefore requires at least two parts), and iteratively increase $N_{\mathcal{P}}$ until a satisfying decomposition has been found.

For our scenario, we have two requirements for an atomic element-wise decomposition: (i) we want to explicitly control the number of segments and (ii) the resulting segments should be connected. We perform a decomposition similar to variational shape approximation (VSA) [15] over our atomic elements. In VSA, each region is represented by a pair of triangle and direction. This triangle serves as a seed for growing the region *as flat as possible* with respect to the provided direction. We modify VSA to take moldability criteria into consideration, and each region represents a part in our decomposition. For a decomposition with $N_{\mathcal{P}}$, we first select $N_{\mathcal{P}}$ elements randomly, and for each region r, we initialize the parting direction $\boldsymbol{d}_r$ with $\boldsymbol{n}_t$.

We then grow regions iteratively by assigning elements with a minimum cost:

$$C(t,r) = \begin{cases} -|\boldsymbol{n}_t \cdot \boldsymbol{d}_r| + \delta & t \text{ belongs to a thin feature} \\ -\boldsymbol{n}_t \cdot \boldsymbol{d}_r + \delta & \text{otherwise,} \end{cases}$$

$$\delta = \begin{cases} 0 & t \text{ does not have overlap with } r \\ w_O & \text{otherwise.} \end{cases}$$

$C(t,r)$ represents the cost for adding an element t to a region r, where $\boldsymbol{d}_r$ is the parting direction for region r, and $\boldsymbol{n}_t$ denotes its outer surface triangle normal.

Intuitively, this energy measures how flat a triangle is while taking overlaps into consideration. For outer triangles belonging to thin features, we allow flips because almost half of the outer triangles of thin features touch the top mold and the others touch the bottom mold (see the thin feature and mold pieces in Fig. 3.6).

Once all elements are assigned to a region, we find new seeds for the next iteration. We first calculate the best direction for each region by computing $\arg\min_{\boldsymbol{d}\in\mathcal{D}} \sum_{t\in r} C(t,r)$. Then we compute best seed triangle for each region by computing $\arg\min_{t\in T_r} C(t,r)$, where T_r is a set of triangles assigned to a region r. If the seeds are unchanged or we reach an upper limit of iterations (we limit the number of iterations up to 20 times as Cohen-Steiner *et al.* [15] suggested), we finish the coarse decomposition.

During coarse decomposition, users can add no-cut design constraints (§3.5) with a brush-like interface.

A no-cut constraint is given as a set of edge-connected triangles u_i over the outer surface, and all atomic elements whose top triangles belong to u_i will be assigned to the same region. To ensure this, we treat the set of atomic elements corresponding to u_i as a single element for region growing. The cost for assigning an atomic element whose outer triangle $t \in u_i$ to a region r is calculated as $\sum_{t_i\in u_i} C(t_i,r)$, and when assigning to a region, all other atomic elements corresponding to u_i are assigned simultaneously.

As output, we obtain $N_{\mathcal{P}}$ parts. Then, for each part we compute an optimal parting line.

3.4.4 Parting Line Computation

Because our targeted fabrication technique is two-piece molding, we need to classify the surface of a part P_i into two connected patches, M_i^+ and M_i^- (blue and orange in Fig. 3.8). While a tempting naive solution would be to simply assign all outer surfaces of our elements to M_i^+, this approach would fail, for example, in thin areas. With the naive solution, the blue patch has a large overhang (red) and this would cause large deformations. With our solution, the undercut caused by T_{in} is removed by adding some material (green) inside.

Therefore, we propose to perform a graph cut-based classification over the surface of each part. The energy terms for the graph cut are

$$E_{\textit{unary, upper}}(t) = \begin{cases} 0 & t \in T_{\text{in}} \\ 1.0 + \boldsymbol{n}_t \cdot \boldsymbol{d} & \text{otherwise} \end{cases}$$

$$E_{\textit{unary, lower}}(t) = \begin{cases} 0 & t \in T_{\text{in}} \\ 1.0 + \boldsymbol{n}_t \cdot (-\boldsymbol{d}) & \text{otherwise} \end{cases}$$

$$E_{\textit{binary}}(t_0, t_1) = \max(0, \boldsymbol{n}_{t_0} \cdot \boldsymbol{n}_{t_1}),$$

where $\boldsymbol{n}_t$ is the normal vector of triangle t and $\boldsymbol{d}$ is the corresponding parting direction for P_i. For unary terms $E_{\textit{unary, upper}}$, $E_{\textit{unary, lower}}$, we do not care if $t \in T_{\text{in}}$ causes some undercuts because the inside is completely invisible after assembly, and we can add some material to remove such undercuts. Adding some material and making the shell thicker also helps liquid material flow during fabrication. For the binary term $E_{\textit{binary}}$, we encourage the boundary between M_i^+ and M_i^- to align to the edges with sharp dihedral angles (e.g., part boundary) for suppressing the visual artifacts caused by parting lines.

Recall that we need a single connected component for M_i^+ and M_i^- because of the nature of two-piece molding. However, this graph cut does not guarantee each segment is a single connected component, which occurs when there are remaining undercuts because of the relaxed moldability constraints. Therefore, we tweak the classification after the graph cut and ensure that both of M_i^+ and M_i^- are single connected components (Fig. 3.9). After the graph cut, each triangle has its label for the classification (top first column). We first update their labels according to whether triangles have overlap. Specifically, we have four types of labels (i) M_i^+ without overlap (orange), (ii) M_i^+ with overlap (red), (iii) M_i^- without overlap (light blue), and (iv) M_i^- with overlap (blue) (top second column).

Second, over connected components of these labels, we construct a graph with asymmetric consts for graph edges (bottom 2nd column).

$$w(c_0, c_1) = \begin{cases} 0 & \text{if both } c_0 \text{ and } c_1 \text{are assigned} \\ & \quad \text{to the same segment } (M_i^+ or M_i^-) \\ |p_{\boldsymbol{d}}(c_1)| & \text{otherwise}\,, \end{cases}$$

where $p_{\boldsymbol{d}}(c)$ is the projected area of connected component c. We iteratively change the label of these nodes and finally obtain a single connected components for M_i^+ and M_i^-. In this graph, we first find the largest M_i^+ component and M_i^- component in terms of projected area (top and bottom nodes), and fix the labeling of these nodes. Then, in projected area's decreasing order, we compute the shortest path from the largest M_i^+ (orange) and M_i^- (light blue). If the path from the largest M_i^+ is shorter, we change the labels of all nodes along the path to M_i^+. Otherwise, we change to M_i^-. For example, in Fig. 3.9, path from the largest M_i^-

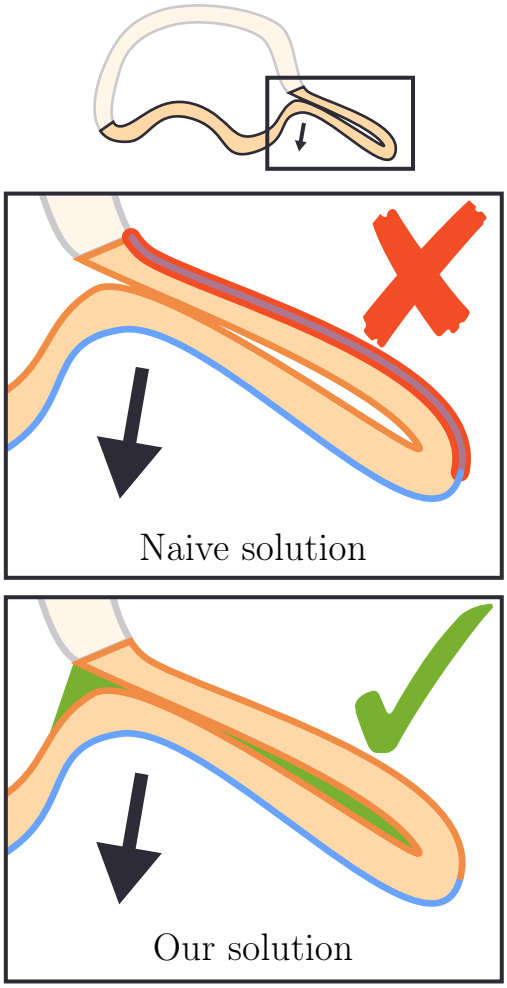


Figure 3.8: Example that naive solution for surface classification does not work.

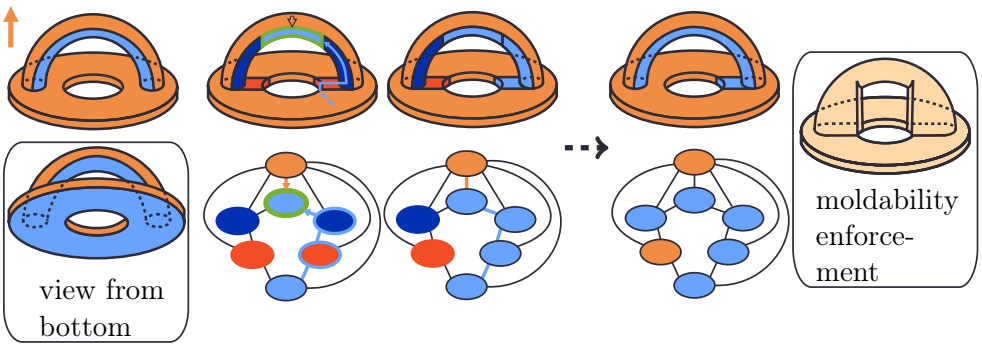


Figure 3.9: Surface classification and moldability enforcement. The part consists of a half solid torus glued to a flat ring. M^- (light blue) is not single connected. We iteratively change the labeling and finally obtain a single connected component for M_i^+ and M_i^-.

is shorter and all labels of nodes along the path are changed to M_i^-. In addition, the labels along the path are fixed and every time the labeling are updated, the costs for the graph edges are also updated. This operation is repeated until the labels of all nodes are fixed.

3.4.5 Part Boundary Smoothing

The boundaries of the parts after region growing follow a set of edges of $\mathcal{S}_{\text{low}}$. Because of the discrete representation, they are usually jaggy and therefore reflect undesired aesthetic and suboptimal moldability conditions (see §3.4.3). We improve the solution by representing the boundary between parts as an active contour model, also called snakes, and perform moldability-aware smoothing and local optimization of the snakes. This approach, as demonstrated by our results, provides aesthetic and molding-friendly smooth boundaries.

We use the same definition as that utilized by Bischoff *et al.* [11] to define the boundaries. Snakes are constituted by a set of snake vertices that lie on the edges or vertices of $\mathcal{S}_{\text{low}}$ and by a set of snake edges connecting snake vertices. The position is represented as weighted sum $\alpha \boldsymbol{v}_0 + (1-\alpha)\boldsymbol{v}_1$, where $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ are the endpoints of a mesh edge. To decompose the shelled object, we add the corresponding points on T_{in} as $\alpha f_V(\boldsymbol{v}_0)+(1-\alpha)f_V(\boldsymbol{v}_1)$ and subdivide the triangles in T_{low} and T_{in} accordingly. Additionally, we also triangulate the sides.

Objectives

The two moldability constraints given by Eq. 3.1 and Eq. 3.2 are hard constraints. Similar to Herholz *et al.* [26], we relax these constraints and quantify their violation with a smooth *moldability energy* function.

Formally, we cast the boundary optimization as a minimization problem,

$$\underset{\text{snakepos}}{\arg\min}\, E = \sum_{i=1}^{N_{\mathcal{P}}} E_P(P_i) + w_l E_{\text{smooth}}\,, \tag{3.3}$$

where $E_P(P_i)$ measures the moldability for P_i,E_{smooth} quantifies the smoothness of the boundary, and w_l is used to reflect the users' preferred smoothness of the part boundary.

Moldability Energy

To quantify the violation of the moldability constraints (i.e., estimated amount of deformation), we define the corresponding energy term E_P as

$$E_P(P_i, \boldsymbol{d}_i) = w_H E_H(P_i, \boldsymbol{d}_i) + w_L E_L(P_i, \boldsymbol{d}_i) \tag{3.4}$$

with the w_H-weighted overhang penalization E_H given as

$$E_H(P_i, \boldsymbol{d}_i) = \sum_{t^+ \in M^+} U(t^+, \boldsymbol{d}_i) + \sum_{t^- \in M^-} U(t^-, -\boldsymbol{d}_i)$$

$$U(t, \boldsymbol{d}) = \begin{cases} -\cos\Big(\min\big(\theta+\phi, \pi\big)\Big)|t| & \theta > \frac{\pi}{2} - \phi \;\wedge\; t \notin T_{\text{in}} \\ 0 & \text{otherwise}\,, \end{cases}$$

where θ represents an angle between the normal of the triangle $\boldsymbol{n}_t$ and parting direction d, namely, $\theta = \arccos(\boldsymbol{n}_t \cdot \boldsymbol{d})$. Essentially, we sum the areas $|t| = |p_{\boldsymbol{d}}(t)|$

of triangles t projected along $\boldsymbol{d}$ for all triangles that violate Eq. 3.1. To ensure that the triangles are tilted with more than draft angle ϕ, we shift the angle between $\boldsymbol{n}_t$ and $\boldsymbol{d}$ by ϕ. Additionally, we ignore the inner triangles T_{in} because we can resolve the violations they caused without any visual artifacts, as previously explained.

We use the same strategy to penalize overlaps that violate Eq. 3.2 with the w_L-weighted energy term

$$E_L(P_i, \boldsymbol{d}_i) = \sum_{t_1^+, t_2^+ \in M^+} O(t_1^+, t_2^+, \boldsymbol{d}_i) + \sum_{t_1^-, t_2^- \in M^-} O(t_1^-, t_2^-, -\boldsymbol{d}_i) \,,$$

where the projected area of the overlap of two triangles is penalized. We define the overlaps as

$$O(t_1, t_2, \boldsymbol{d}) = \begin{cases} & t_1 \neq t_2 \\ \frac{1}{2}|p_{\boldsymbol{d}}(t_1) \cap p_{\boldsymbol{d}}(t_2)| & \wedge\, (\boldsymbol{n}_{t_1} \cdot \boldsymbol{d})(\boldsymbol{n}_{t_2} \cdot \boldsymbol{d}) \geq 0 \\ & \wedge\, t_1, t_2 \notin T_{\text{in}} \\ 0 & \text{otherwise} \,, \end{cases}$$

avoiding a second penalization of overhang triangles. In the same manner with E_H, we ignore the triangles in T_{in}. Large w_L and w_H enforce moldability constraints more strictly. Note that for a given set $\mathcal{D}$ of parting directions, the projected areas of both $|p_{\boldsymbol{d}}(t)|$ and $|p_{\boldsymbol{d}}(t_1) \cap p_{\boldsymbol{d}}(t_2)|$ can be precomputed, thus removing these costly computations from the innermost loop of our method.

Regularization

To obtain a aesthetically pleasing boundary, smooth decompositions are often desired. Furthermore, in practice, smooth boundaries are preferred as a general design guideline because jagged and thin parts might break off. To address these design considerations, we use a smoothing energy term

$$E_{\text{smooth}} = \sum_i \frac{(x_{i-1} - x_i) \cdot (x_{i+1} - x_i)}{\|(x_{i-1} - x_i)\| \, \|(x_{i+1} - x_i)\|} \Big/ \#\text{snakes vertex} \,,$$

measuring the average of angles between consecutive line segments at snake vertex positions x_i .

Optimization

As Eq. 3.3 depends on the global geometry configuration, computing an analytic gradient is non-trivial. Instead, we first evaluate a desired displacement direction for each vertex, which we choose as the gradient of the smoothness energy. In this subspace, we compute the gradient of Eq. 3.3 by projecting new snake positions onto the closest surface and using finite differences. We then apply simple gradient descent and recompute the parting lines of the newly obtained parts after each iteration (§3.4.4).

3.5 User Interaction

Before and after the decomposition, users can edit the decomposition by using the following four functions: no-cut, cut, merge, and selective smoothing (Fig. 3.10). Before the decomposition, users can specify sets of edge-connected triangles, and these sets will be fabricated without division. This is achieved

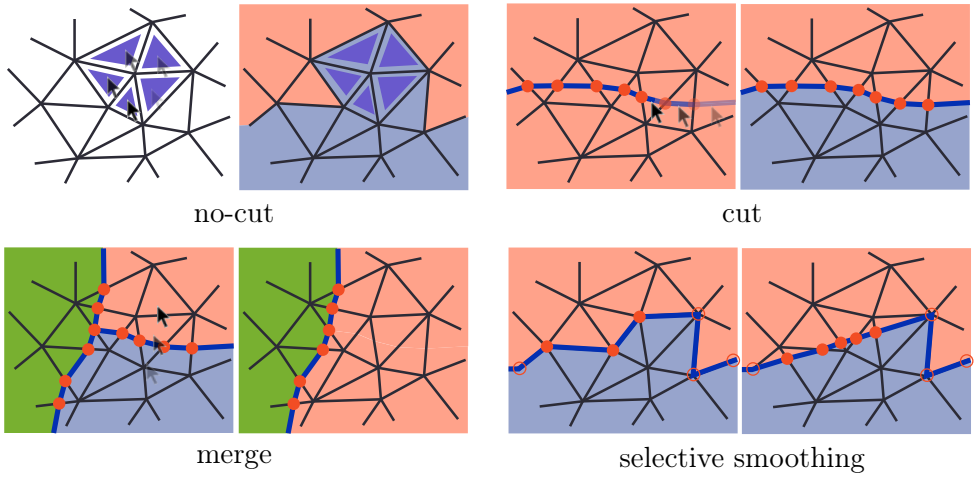


Figure 3.10: During interactive editing, users can edit the decomposition using four functions: no-cut, cut, merge, and selective smoothing.

by assuming that the sets are single elements in region growing, as explained. After the decomposition, users can edit the boundary with the cut, merge, and selective smoothing functions. To cut the parts, we simply introduce new snake vertices and snake edges according to the user input. After cutting the parts, we update the corresponding parting directions for each divided part by computing $\arg\min_{\boldsymbol{d}\in\mathcal{D}} E_P(P_i, \boldsymbol{d})$ with fixed part P_i. To merge parts p_0 and p_1 that are neighbors, we compute the new part $p_2 = p_0 \cup p_1$ and the best parting direction by $\arg\min_{\boldsymbol{d}\in\mathcal{D}} E_P(P_2, \boldsymbol{d})$. Users can apply the smoothing to user-selected vertices to increase smoothness locally. During user interaction, the moldability energy is tracked. When it exceeds the threshold γ, the system provides a warning. Furthermore, the estimated amount of deformation is visualized to allow users to see where the shape might be deformed.

3.6 Mold Generation

After the satisfactory decomposition of the thin shelled low-res representation V into a set of parts $\mathcal{P}$ was achieved, the corresponding molds can be automatically generated. Before actual fabrication (see §3.6.3), we still need to correct the following three issues that stem from the approximations we used to guarantee interactivity: (i) useless inner triangles of thin features still remain inside of the solid hull of thin features, (ii) surface details were lost during the remeshing (see §3.4.2), and (iii) because of the relaxation of the moldability constraints (see Eq. 3.4), small overhangs or global overlaps can be present along the parting directions. We solve these issue in the same order and start by obtaining surface meshes that fully enclose thin parts. After performing detail restoration (see §3.6.1), we eliminate all remaining violations of the moldability constraints by using swept volumes (see §3.6.2).

3.6.1 Detail Restoration

To restore the fine details of the input surface $\mathcal{S}_{\text{input}}$, which were lost during the initial remeshing for generating low-res representation, we augment the outer surface (i.e., $\mathcal{S}_{\text{low}}$) with the original surface geometry. As illustrated in Fig. 3.11, we use a two-step process. First, the outer surface $\mathcal{S}_{\text{low}}$ of each part P_i is offset outwards such that the corresponding patch of $\mathcal{S}_{\text{input}}$ is fully contained in the thereby thickened part $\widehat{P}_i$. By computing the intersection of $\mathcal{S}_{\text{input}}$ and $\widehat{P}_i$, we obtain a high-detail replacement for the outer surface $\mathcal{S}_{\text{low}}$. Note that the interior surface is kept as is because it is generally not visible in the assembled model. In the remainder of this section, we use P_i to refer to this *replacement*.

3.6.2 Moldability Enforcement

As we relaxed the moldability constraints and as the outer mold face was replaced with a different geometry, we cannot guarantee that the individual parts P_i of the decomposition are strictly moldable; minor overhangs or global overlaps could still persist. To eliminate all such artifacts, we follow a similar strategy as that used by Herholz *et al.* . [26]. Because our fabrication is two-sided, we apply the moldability enforcement to M_i^+ and M_i^- independently. We first try to resolve most violations (i.e., overhangs and overlaps) by an as-rigid-as-possible (ARAP) [62] based deformation. The remaining violations are addressed using swept volumes. We compute swept volumes along the parting directions $\boldsymbol{d}_i$. As illustrated in

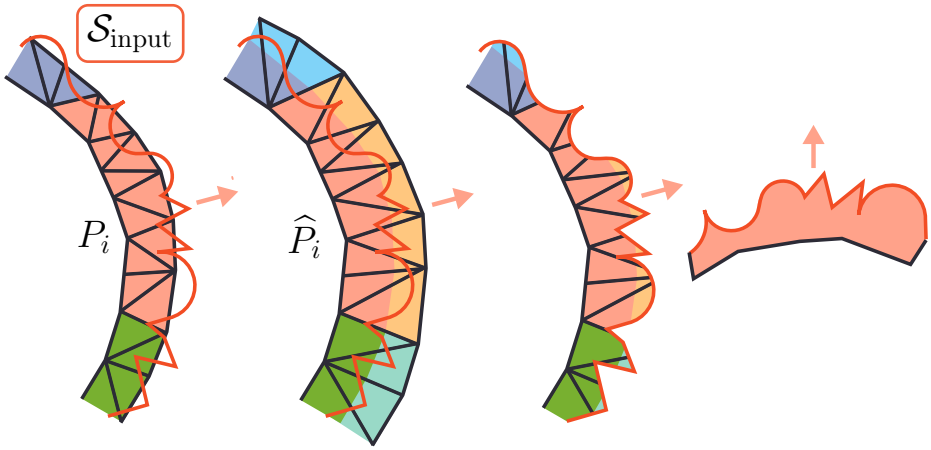


Figure 3.11: Detail restoration. A part P_i resulting from the decomposition of the low-resolution representation $\mathcal{S}_{\text{low}}$ (left) is offset to contain the original high-resolution surface $\mathcal{S}_{\text{input}}$ (center). A high-detail replacement of the outer surface P_i is obtained by intersecting both.

Fig. 3.12, each of the two mold faces, M_i^+ and M_i^-, can be swept along its corresponding parting directions $\boldsymbol{d}_i$ and $-\boldsymbol{d}_i$ or along the opposite directions. In case swept volumes are used, the potential intersections of the added volume with other object parts need to be checked and removed. The two resulting swept volumes SV_i^{+-} and SV_i^{-+} can be combined using boolean operations to yield fabricable mold pieces.

3.6.3 Sprues, Runners, and Design Finishing

With the aforementioned pieces, we assembled the mold geometry by considering the properties of the casting process. As shown in Fig. 3.2, the assembled mold requires in- and outlets to allow for resin insertion and avoid the creation of air pockets. In addition, mold geometries have two requirements: (i) all parts need to be connected by runners (channel structures for material flow) and gates (connection between parts and runners), and (ii) the touching surface between the two mold pieces should exactly pass through the parting lines of all parts. To achieve such a mold geometry, we begin with arranging all parts so that their parting directions become the +Z direction. The smooth surfaces between the parting lines and the outer borders of the molds are generated by interpolating the height field with the use of radial basis functions with a Gaussian kernel and vertices on the parting lines as constraints [12]. For runners, we use simple pre-defined structures. For the gates, we simply place them at the lowest and locally highest points in the Y-coordinates on the parting line for each part and connect them with runners. Finally, we compute mesh boolean $(B \cup (\bigcup SV^{+-})) \setminus (\bigcup SV^{-+})$ for the bottom and top molds. In practice, mesh boolean for mold geometry generation and moldability enforcement can be performed concurrently.

In our system, users can make a common mold for all the parts (see the bunny and airplane model in Fig. 3.18), or separate molds for each part (see the others in Fig. 3.18). A large common mold reduces the effort for fabrication (i.e., manual casting). On the other hand, separate molds allow the fabrication of each part with a different material and the creation of a larger object by maximally scaling up the mold pieces to the printing volume of a 3D printer.

The final mold geometry can be realized using either 3D printing or milling. As the individual mold pieces are height fields, supporting structures are not required for 3D printing. Moreover, conventional three-axis milling – if the diameter of the milling bit and head allow – can also be used. We utilize photopolymerization based 3D printing to fabricate all mold pieces shown in the chapter.

3.7 Results

We used our system to decompose a variety of shapes and compute the corresponding molds; a detailed overview of our results is given in Fig. 3.16 and Fig. 3.18. For all examples, all models are scaled to fit into a unit cube, and we set the weight for penalizing overhangs and overlaps $w_H = 1.0, w_L = 1.0$, the weight for penalizing overlaps during the coarse decomposition $w_O = 10^6$, $w_l = 1.0$, the thickness of the shell $\omega = 0.03$, the draft angle $\phi = \frac{\pi}{180}$, and the user-defined threshold for the amount of deformation $\gamma = 0.05$. The set $\mathcal{D}$ of possible parting directions is given by the vertex coordinates of an icosphere with three subdivisions (162 vertices). Because of the symmetry of the icosphere, storing only one hemisphere of it and obtaining all remaining directions of $\mathcal{D}$ by mirroring is sufficient.

Statistics on the results, including mesh complexity and timings can be found in Tab. 3.2 and Tab. 3.3. For the precomputation of triangle–triangle overlaps

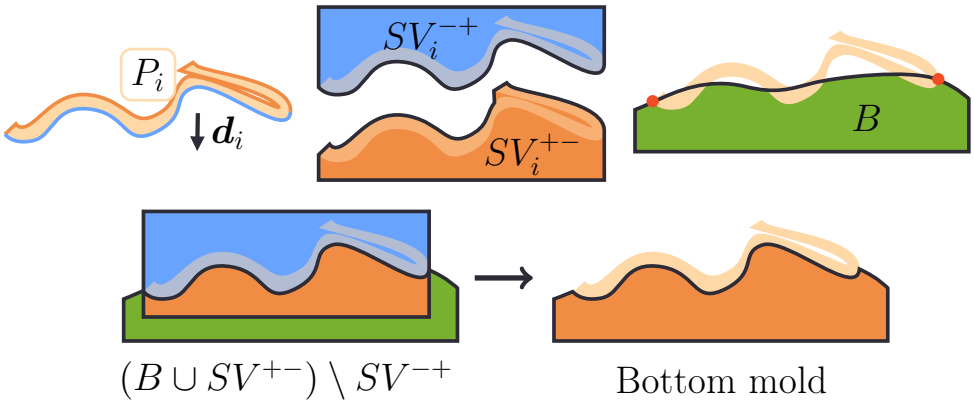


Figure 3.12: Sweeping along $\boldsymbol{d}_i$ and $-\boldsymbol{d}_i$ removes the remaining undercut. In addition, we also compute the smooth touching surface (B) that passes through the parting line (red circle). The final mold geometries are generated by mesh boolean with these volumes.

Table 3.2: Result statistics. The timing was measured on an Intel Core i7 (2.2GHz) with 16GB RAM with single thread.

	# triangles ($\mathcal{S}_{\text{input}}$ / $\mathcal{S}_{\text{low}}$)	pre-comp. for coarse decomp. per direction	coarse decomp. per iteration	# parts
Sphere	20480 / 320	3.84 ms	11.0 ms	2
Torus	20000 / 1152	34.9 ms	82.0 ms	2
Kitten	39206 / 2310	136 ms	396 ms	2
Airplane	119866 / 19136	13910.28 ms	31711 ms	2
Fertility	51534 / 8788	2011.67 ms	4886.17 ms	2
Bunny	49700 / 2740	210 ms	2110 ms	2
Sculpture	72006 / 5888	875.25 ms	2550 ms	6
Beethoven	419424 / 2000	118.62 ms	1446 ms	7

Table 3.3: Result statistics about user interactions. The timing was measured on an Intel Core i7 (2.2GHz) with 16GB RAM with single thread. Several interactions were not used for generating the examples (e.g., sphere did not need no-cut, cut, and merge interactions), but we measured the calculation times as a reference. For smoothing, we measured the time for applying smoothing to all the snakes vertices.

	# no-cut	cut (# / timitng)	merge (# / timitng)	smoothing (timing)
Sphere	0	0 / 5.60 ms	0 / 26.2 ms	90.8 ms
Torus	0	0 / 15.0 ms	0 / 71.4 ms	665 ms
Kitten	0	0 / 47.4 ms	0 / 209 ms	3964 ms
Airplane	0	0 / 692 ms	0 / 1872 ms	176574 ms
Fertility	0	1 / 208 ms	1 / 891 ms	51580 ms
Bunny	0	0 / 53.2 ms	0 / 232 ms	3430 ms
Sculpture	0	8 / 415 ms	8 / 820 ms	40633 ms
Beethoven	1	5 / 210 ms	0 / 648 ms	18973 ms

for coarse decomposition, we use early rejection through axis-aligned bounding boxes. Thus, the actual number of projected triangle overlaps is the dominant factor with regard to runtime, which is in the order of seconds for all our low-res meshes $\mathcal{S}_{\text{low}}$. Additionally, column four to seven report statistics on the user interactions, including the number of no-cut, cut, and merge operations, as well as the timing on performing these individual operations.

We started testing our system with simple shapes, such as sphere and torus (Fig. 3.16). These examples enable us to intuitively validate the quality of the decomposition. For a sphere, our system found a $N_{\mathcal{P}} = 2$ parts decomposition with two nearly perfect half-spheres, as expected. We obtained a similar result for a torus.

The boundary optimization method provides appealing and smooth boundaries in just a few steps. (Fig. 3.15) shows the fast decay of the smoothness energy for the bunny model, starting from an automatically obtained coarse initialization, while the moldability energy E_P stays approximately constant. In this example, the moldability energy is lower than the user-defined threshold $\gamma = 0.05$ (green), and this decomposition is acceptable.

A key feature of our method is its ability to be applicable to shapes with non-zero genus (see the torus, kitten, fertility, and sculpture examples). Furthermore, thin parts are successfully detected and suitable molds for their fabrication as solid pieces generated (see the wings of airplane, arms of the fertility model, and ears of the bunny). Since our method provides the user with the ability to specify the location of part boundaries, it is possible to create semantically meaningful moldable parts. As example, the Beethoven model was successfully decomposed into parts that reflect the coloring of the original model. We compared the smooth RBF surface with a naive solution, where only a flat surface was used (Beethoven). This results in more challenging borders, which are more fragile and difficult to handle in practice.

Decompositions of the sphere, torus, kitten, airplane, and bunny examples were computed fully automatic. For the fertility and sculpture example, the automatically generated decomposition was similar with the result shown in Fig. 3.18, but minor user interactions with our system were required to obtain the visually pleasing results. For the Beethoven example, our system generates an initial decomposition with fewer parts, as shown in the accompanied video. However, our interface allows users to intuitively split the initial solution into additional parts for fabricating them with different color. In current implementation, detected thin features are automatically labeled as no-cut regions.

For example, the arms of the Fertility model are automatically labeled as no-cut regions, and there were no additional user defined no-cut regions required. For a demonstration of the interactive design system we refer to the accompanied video.

In addition, our system visualizes the deformation estimate in real time, allowing users to identify potentially problematic regions (colored red) and eventually adapting their design. We also fabricated most of our models, using either resin casting or injection molding.

Casting The casting process begins by covering each part of the mold with a release agent to facilitate demolding of the cured pieces. Because of the generated runners and gates and because the mold is not perfectly airtight at parting lines, we did not encounter problems with trapped air in practice. For the result shown in Fig. 3.18, we use ultra-low viscose urethane casting resins with a high shore

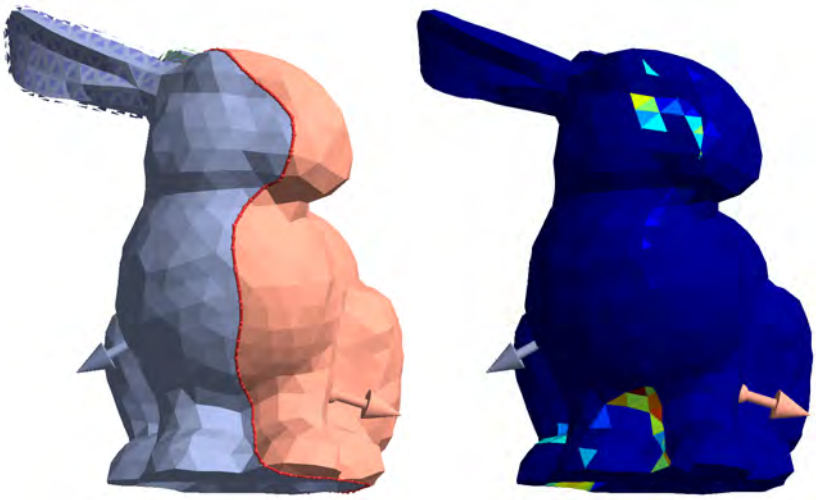


Figure 3.13: The visualization of the estimated amount of deformation in our system. The estimated amount is visualized by using heatmap, and blue indicates smaller deformation and red indicates larger deformation. As expected, the deformation is mainly concentrated around the boundary of neighboring parts.

and a pot life of seven minutes for the Beethoven example and five minutes for the others. The manually placed pins serve the sole purpose to easily open the molds. The resulting piece may have some imperfections along the parting line, such as a thin *molding flash*, a typical artifact caused by the leakage of the material between the two touching surfaces of the mold. This excess material is removed using a rasp. Finally, the casted pieces are glued together along the seams to complete the target model.

Injection Molding We also tested one of our molds on a professional industrial injection molding machine. Using our system, we decomposed the Stanford bunny into two shell pieces and generated a single two-piece mold. No adjustments to our workflow were required, except for defining the position of the sprue bush to prevent the shell pieces from colliding with it. Manual post-processing was limited to drilling holes for fixing the mold to the machine and for inserting the cylindrical metal sprue bush. We waived the addition of a mechanism for automatic ejection because of the low production volume. However, adding ejector pins would be straight forward and only require the placement of a few cylindrical holes. The injection molded samples are made of polypropylene with a material consumption of 26g for the actual model and 5g for the sprue. The material cost per sample was approximately 10 cents.

Comparison to [26] Herholz et al. [26] presented a method for decomposing and deforming a surface mesh into height-field patches. The targeted fabrication technique, casting with multi-mold pieces, significantly differs from our two-piece molding. However, their method could be re-purposed for generating thin shell objects for two-piece molding. For highlighting the difference to [26], we computed decompositions of several models with both methods. For computing the decompositions of thin shell objects based on [26], we first segment the surface with [26], and then extrude the patches inward to obtain thin shell parts. Finally, we compute a parting line for each part, and apply moldability enforcement to ensure moldability.

For the comparison shown in Fig. 3.17, we used the same value for the parameter γ that corresponds to the maximally allowed amount of deformation. Because [26] does not allow overlaps even if they could be solved by the moldability enforcement (i.e., deformation) and furthermore does not allow two-sided parts (e.g., the arms of the fertility), their method results in a larger number of parts. Specifically, [26] decomposes the kitten model into five parts, fertility into eight parts, and the bust sculpture into six parts. Our method can decompose all the models into two parts. Furthermore, we evaluate the difference from the original shape by computing the Hausdorff distance from original shape to the deformed shell parts. Fig. 3.17 also shows the ratio of the distance to the diagonal of the bounding box of the shape. In all comparisons, results generated with our method have less deformation than [26]. Additionally, the decompositions of the airplane highlight the effect of our volume-awareness on the decomposition result. As shown in Figure Fig. 3.14, the method of Herholz et al. [26] generates very thin and fragmented parts (e.g., the wings are splitted into thin flat parts) because their method is not aware of thin features. On the other hand, our method keeps the wings intact, which improves both aesthetics and stability (Fig. 3.18).

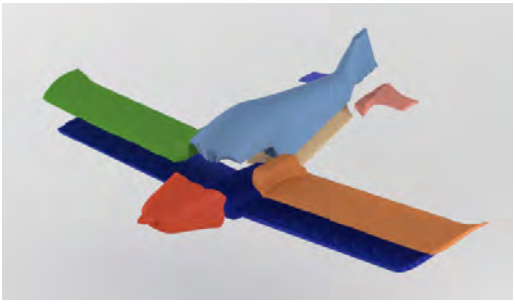


Figure 3.14: Decomposition of airplane by [26]. Their decomposition algorithm is not aware of the thin features, the object is fragmented and its wings are splitted into two very thin parts.

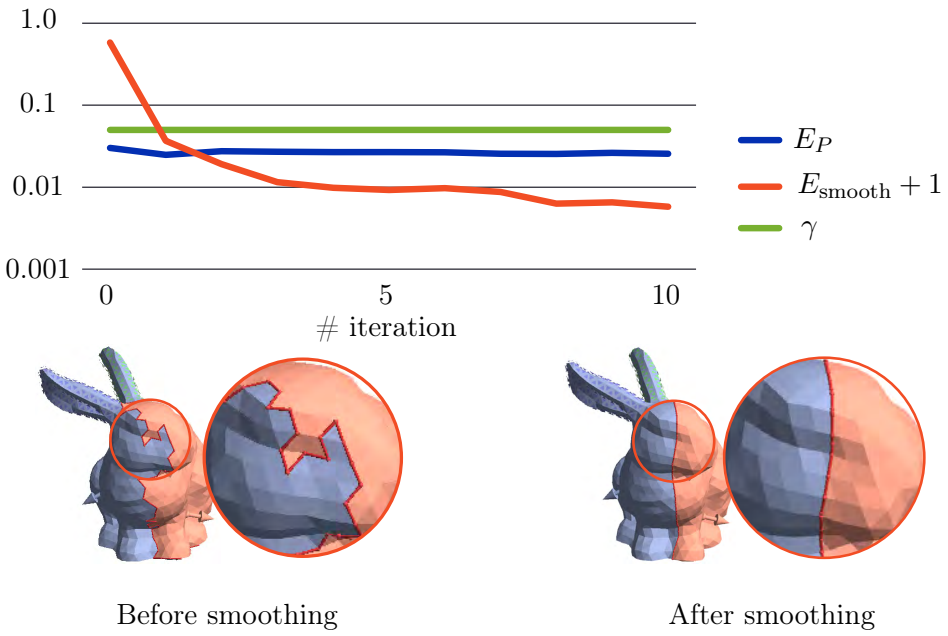


Figure 3.15: Energy during boundary smoothing. The smoothness energy (red) sharply decreases in the first few iterations and then converges around an optimal solution. The moldability energy E_P is shown in blue.

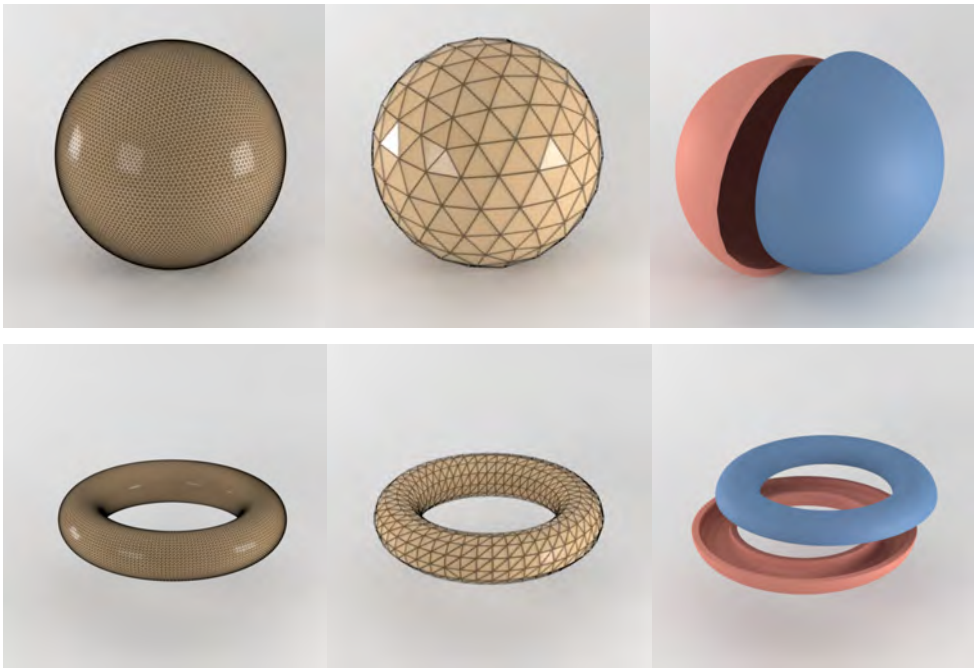
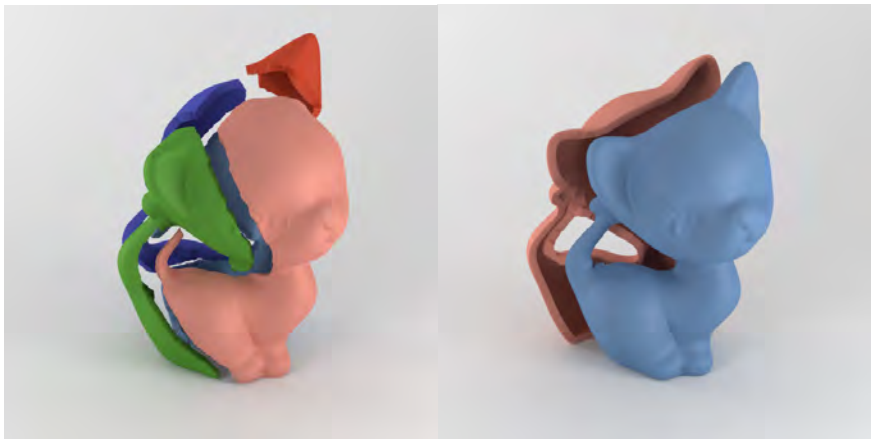


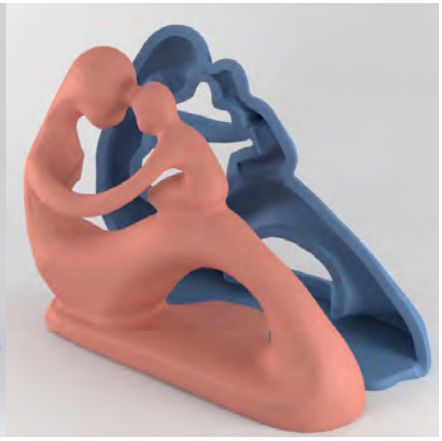
Figure 3.16: Test with simple sphere and torus. (From left to right) High-res detailed input $\mathcal{S}_{\text{input}}$, low-res representation $\mathcal{S}_{\text{low}}$, final decomposition.



max 1.97% avg 0.10%

max 0.44% avg 0.07%

max 4.78% avg 0.48%



max 1.97% avg 0.02%

max 1.16% avg 0.06%

[26]

max 1.00% avg 0.04%

Our method

Figure 3.17: Side-by-side comparison of [26] and our method. From top to bottom: Kitten, fertility, and bust sculpture. The average and maximal Hausdorff distances to the input mesh are given as ratio with respect to the model's bounding box diagonal.

Figure 3.18: (From left to right) High-res detailed input $\mathcal{S}_{\text{input}}$, low-res representation $\mathcal{S}_{\text{low}}$, final decomposition, corresponding automatically generated mold pieces, and fabrication results. (From top to bottom) We denote our examples as kitten, airplane, fertility, bunny, sculpture, and Beethoven. Each pair of mold pieces corresponding to the part with the same color. Bunny example is designed for injection molding, and the others are designed for manual resin casting. For the Beethoven model, we omit the computation of the smooth surface during the mold generation.

3.8 Limitations

Although our method can deal with a wide range of shapes, it inevitably has some limitations. With regard to the position of cutting seams, our energy formulation for active contours only considers moldability and smoothness, and further aesthetic considerations are left to the users. Therefore, the contours might sometime move in a subjectively undesired direction. This issue could be limited by using perceptual models that quantify the visual quality of a decomposition [38, 57, 76].

The current placement of each part on the mold is a simple rotation so that its parting direction points upward along +Z. An obvious extension would be an optimal packing, which we left for future work. Another interesting avenue for future work would be to extend our system to take the flow and the temperature of the material into account. Especially for geometrically challenging cavities and for optimizing economic factors, flow and heat dissipation are important performance indicators.

3.9 Summary

In this chapter, we propose an interactive decomposition method for two-piece molding. The proposed method solves a computationally complex problem (i.e., moldability check) in interactive speed by reducing the complexity of the target shape (coarse-to-fine approach) and relaxing the constraints for moldability. The method successfully decomposes a wide variety of shapes as shown in Fig. 3.16 and Fig. 3.18. To demonstrate the applicability of our method for hobbyist makers and in an industrial setting, we fabricated several physical copies of popular models in computer graphics with resin casting and injection molding.

Chapter 4

Drain Hole Placement Optimization for Powder Recovery

This chapter is removed due to its confidential content.

Chapter 5

Conclusion

In this chapter, we conclude this thesis. First, we summarize our contribution and then discuss the limitation and the potential extension of our methods. Finally, we illustrate future directions of the researches related with fabrication.

5.1 Summary of Our Contributions

In this thesis, we tried to relieve the manual effort for digital fabrication process. The manual effort is mainly spent during the pre-processing and the post-processing for the fabrication process. As we explained in the §1, there are various kinds of fabrication techniques that are commonly used in the actual fabrication. Consequently, the manual effort varies widely depending on the fabrication technique employed. Furthermore, to my best knowledge, there are no general solution for these varying manual efforts. So, during my graduate study, we try to solve specific two manual efforts; 1) pre-processing for fabrication with molding, and 2) pre- and post-processing for fabrication with powder-type 3D printing.

1) is decomposing a complex shape into multiple simpler shapes in order to fabricate traditional molding technique. This problem requires optimizations of not only decomposition, but also the parting direction (i.e. the direction that mold pieces moves during the dis-assembly). For efficiently solve this problem we optimize them alternative manner. We also presented simple yet practical mold geometry generation algorithm applied after optimizing the decomposition. Finally, we demonstrate several results including very challenging examples, and which shows potential of our method.

This paragraph is removed due to its confidential content.

5.2 Limitations

As we discussed above, both of our methods have several limitations. Here we discuss the limitations from higher perspective. As we mentioned several times, our methods focus on very specific fabrication methods and specific manual effort. Thus, even though our methods solve each problem successfully, the applicability is still limited. However, as we noted in §1, due to the variation of the fabrication techniques, we think that this limitation is inevitable.

In addition to the limitation above, our method does not adapt the users' preference. For example, there are studies that employed some human computation techniques to reflect the users' preference [34, 35, 36]. Specifically, Koyama et al. [35] employed a progressive learning technique and that enables to reflect the users' preference at runtime by referring the editing history. It is expected

that a user uses our system continuously, and thus integrating some learning algorithm and reflecting users' preference might greatly increase the usability of our methods.

Further, current interface is not yet completely polished. Thus, we would perform further user study and have discussions with potential users (from novel users to professional artists). Through the study and the feedback, we would find some design principle for our scenario. If needed, we consider that we should implement our method as plugins for some existing 3D modeling software, such as Blender or ZBrush.

5.3 Future Directions

One straightforward future direction of my graduate study is relieving the other fabrication techniques. For example, removing support structure after printing with typical fused deposition modeling (FDM) is very time consuming and takes much human effort to finish. Furthermore, finishing by using sanders also take much effort. When we have much *smart* structure for support, it is expected to reduce the time and effort for FDM printers. In addition, typical stamping is a very traditional and efficient fabrication technique but it only allows to fabricate very simple shape. Similar with the decomposition for molding we described above, there are possibilities to enhance the applicability of stamping. Furthermore, there are various kinds of fabrication techniques, for instance, laser cutting, knitting, and milling. Thus, there are very large space need to be explored.

The direction I describe above is enhancing the existing fabrication technique by relieving the cost (time and human effort) within the fabrication process. This direction is very important and would have large impact on the industry. In parallel with this practical direction, I would explore another direction. In very recent years, the fabrication technique become well developed, and there are several researches that enables to fabricate very sophisticated objects. For instance, Auzinger et al. [5] fabricates thin plate with structural color by using nano-scale 3D printer. To fabricate such sophisticated objects, people need to maximize the potential of the fabrication technique and which cannot be done without optimization technique. So, the other direction is to develop algorithms that maximally exploit the fabrication technique and realize very sophisticated objects that cannot be realized without algorithms.

References

[1] Thomas Alderighi, Luigi Malomo, Daniela Giorgi, Bernd Bickel, Paolo Cignoni, and Nico Pietroni. Volume-aware design of composite molds. *ACM Transactions on Graphics*, 38(4):1–12, jul 2019.

[2] Thomas Alderighi, Luigi Malomo, Daniela Giorgi, Nico Pietroni, Bernd Bickel, and Paolo Cignoni. Metamolds: Computational design of silicone molds. *ACM Transactions on Graphics*, 37(4):1–13, jul 2018.

[3] Giuseppe Alemanno, Paolo Cignoni, Nico Pietroni, Federico Ponchio, and Roberto Scopigno. Interlocking pieces for printing tangible cultural heritage replicas, 2014.

[4] Marc Alexa, Kristian Hildebrand, and Sylvain Lefebvre. Optimal discrete slicing. *ACM Transactions on Graphics*, 36(4):1, jan 2017.

[5] Thomas Auzinger, Wolfgang Heidrich, and Bernd Bickel. Computational design of nanostructural color for additive manufacturing. *ACM Transactions on Graphics*, 37(4):1–16, jul 2018.

[6] Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. Fabricating articulated characters from skinned meshes. *ACM Transactions on Graphics*, 31(4):1–9, jul 2012.

[7] Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. Spin-it: optimizing moment of inertia for spinnable objects. *ACM Transactions on Graphics*, 33(4):1–10, jul 2014.

[8] Dustin Beyer, Serafima Gurevich, Stefanie Mueller, Hsiang-Ting Chen, and Patrick Baudisch. Platener: Low-fidelity fabrication of 3d objects by substituting 3d print with laser-cut plates. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*. ACM Press, 2015.

[9] Gaurav Bharaj, Stelian Coros, Bernhard Thomaszewski, James Tompkin, Bernd Bickel, and Hanspeter Pfister. Computational design of walking automata. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation - SCA '15*. ACM Press, 2015.

[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 Transactions on Graphics*, 29(4):1, jul 2010.

[11] Stephan Bischoff, Tobias Weyand, and Leif Kobbelt. Snakes on triangle meshes. In *Bildverarbeitung für die Medizin 2005*, pages 208–212. Springer-Verlag, 2005.

[12] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*. ACM Press, 2001.

[13] Pritam Chakraborty and N. Venkata Reddy. Automatic determination of parting directions, parting lines and surfaces for two-piece permanent molds. *Journal of Materials Processing Technology*, 209(5):2464–2476, mar 2009.

[14] Xuelin Chen, Hao Zhang, Jinjie Lin, Ruizhen Hu, Lin Lu, Qixing Huang, Bedrich Benes, Daniel Cohen-Or, and Baoquan Chen. Dapper: decompose-and-pack for 3d printing. *ACM Transactions on Graphics*, 34(6):1–12, oct 2015.

[15] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905, aug 2004.

[16] Chengkai Dai, Charlie C. L. Wang, Chenming Wu, Sylvain Lefebvre, Guoxin Fang, and Yong-Jin Liu. Support-free volume printing by multi-axis motion. *ACM Transactions on Graphics*, 37(4):1–14, jul 2018.

[17] Jérémie Dumas, Jean Hergel, and Sylvain Lefebvre. Bridging the gap: automated steady scaffoldings for 3d printing. *ACM Transactions on Graphics*, 33(4):1–10, jul 2014.

[18] Oskar Elek, Denis Sumin, Ran Zhang, Tim Weyrich, Karol Myszkowski, Bernd Bickel, Alexander Wilkie, and Jaroslav Křivánek. Scattering-aware texture reproduction for 3d printing. *ACM Transactions on Graphics*, 36(6):1–15, nov 2017.

[19] Jimmy Etienne, Sylvain Lefebvre, Nicolas Ray, Daniele Panozzo, Samuel Hornus, Charlie C. L. Wang, Jonàs Martínez, Sara McMains, Marc Alexa, and Brian Wyvill. CurviSlicer: slightly curved slicing for 3-axis printers. *ACM Transactions on Graphics*, 38(4):1–11, jul 2019.

[20] Sándor P. Fekete and Joseph S. B. Mitchell. Terrain decomposition and layered manufacturing. *International Journal of Computational Geometry & Applications*, 11(06):647–668, dec 2001.

[21] Chi-Wing Fu, Peng Song, Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman, and Daniel Cohen-Or. Computational interlocking furniture assembly. *ACM Transactions on Graphics*, 34(4):91:1–91:11, jul 2015.

[22] J.Y.H. Fuh, M. W. Fu, and A.Y.C. Nee. *Computer-Aided Injection Mold Design and Manufacture*. Plastics Engineering. CRC Press, 2004.

[23] Tsukasa Fukusato, Morihiro Nakamura, and Takeo Igarashi. Interactive design and optimization of free-formed returning boomerang. In *SIGGRAPH Asia 2018 Technical Briefs on - SA '18*. ACM Press, 2018.

[24] Wei Gao, Yunbo Zhang, Diogo C. Nazzetta, Karthik Ramani, and Raymond J. Cipra. RevoMaker: Enabling multi-directional and functionally-embedded 3D printing using a rotational cuboidal platform. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15*. ACM Press, 2015.

[25] Ruslan Guseinov, Eder Miguel, and Bernd Bickel. CurveUps: shaping objects from flat plates with tension-actuated curvature. *ACM Transactions on Graphics*, 36(4):1–12, jul 2017.

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

[27] Ruizhen Hu, Honghua Li, Hao Zhang, and Daniel Cohen-Or. Approximate pyramidal shape decomposition. *ACM Transactions on Graphics*, 33(6):1–12, nov 2014.

[28] Masatomo Inui, Hidekazu Kamei, and Nobuyuki Umezu. Automatic detection of the optimal ejecting direction based on a discrete gauss map. *Journal of Computational Design and Engineering*, 1(1):48–54, jan 2014.

[29] Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics*, 32(4):1, jul 2013.

[30] Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. Instant field-aligned meshes. *ACM Transactions on Graphics*, 34(6):1–15, oct 2015.

[31] David O. Kazmer. *Injection Mold Design Engineering*. Carl Hanser Verlag, 2016.

[32] Rahul Khardekar, Greg Burton, and Sara McMains. Finding feasible mold parting directions using graphics hardware. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling - SPM '05*. ACM Press, 2005.

[33] Robert Kovacs, Anton Synytsia, Patrick Baudisch, Alexandra Ion, Pedro Lopes, Tim Oesterreich, Johannes Filter, Philipp Otto, Tobias Arndt, Nico Ring, and Melvin Witte. TrussFormer: 3D Printing Large Kinetic Structures. In *The 31st Annual ACM Symposium on User Interface Software and Technology - UIST '18*. ACM Press, 2018.

[34] Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. Crowd-powered parameter analysis for visual design exploration. In *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*. ACM Press, 2014.

[35] Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. SelPh. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. ACM Press, 2016.

[36] Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. Sequential line search for efficient visual design optimization by crowds. *ACM Transactions on Graphics*, 36(4):1–11, jul 2017.

[37] Yuki Koyama, Shinjiro Sueda, Emma Steinhardt, Takeo Igarashi, Ariel Shamir, and Wojciech Matusik. AutoConnect: computational design of 3D-printable connectors. *ACM Transactions on Graphics*, 34(6):1–11, oct 2015.

[38] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. *ACM Transactions on Graphics*, 24(3):659, jul 2005.

[39] Weishi Li, R.R. Martin, and F.C. Langbein. Molds for meshes: Computing smooth parting lines and undercut removal. *IEEE Transactions on Automation Science and Engineering*, 6(3):423–432, jul 2009.

[40] Alan C. Lin and Nguyen Huu Quang. Automatic generation of mold-piece regions and parting curves for complex CAD models in multi-piece mold design. *Computer-Aided Design*, 57:15–28, dec 2014.

[41] Lin Lu, Baoquan Chen, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, and Daniel Cohen-Or. Build-to-last: strength to weight 3d printed objects. *ACM Transactions on Graphics*, 33(4):1–10, jul 2014.

[42] Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. Chopper: partitioning models into 3d-printable parts. *ACM Transactions on Graphics*, 31(6):1, nov 2012.

[43] Luigi Malomo, Nico Pietroni, Bernd Bickel, and Paolo Cignoni. FlexMolds: automatic design of flexible shells for molding. *ACM Transactions on Graphics*, 35(6):1–12, nov 2016.

[44] Tobias Martin, Nobuyuki Umetani, and Bernd Bickel. OmniAD: data-driven omni-directional aerodynamics. *ACM Transactions on Graphics*, 34(4):113:1–113:12, jul 2015.

[45] Stefanie Mueller, Sangha Im, Serafima Gurevich, Alexander Teibrich, Lisa Pfisterer, François Guimbretière, and Patrick Baudisch. WirePrint: 3D printed previews for fast prototyping. In *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*. ACM Press, 2014.

[46] Stefanie Mueller, Tobias Mohr, Kerstin Guenther, Johannes Frohnhofen, and Patrick Baudisch. faBrickation: fast 3D printing of functional objects by integrating construction kit building blocks. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, 2014.

[47] Morihiro Nakamura, Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. An interactive design system of free-formed bamboo-copters. *Computer Graphics Forum*, 35(7):323–332, oct 2016.

[48] 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 Transactions on Graphics*, 37(4):1–13, jul 2018.

[49] Kazutaka Nakashima, Yuki Koyama, Takeo Igarashi, Takashi Ijiri, Shin Inada, and Kazuo Nakazawa. Interactive Deformation of Structurally Complex Heart Models Constructed from Medical Images. In T. Bashford-Rogers and L. P. Santos, editors, *EG 2016 - Short Papers*. The Eurographics Association, 2016.

[50] A.Y.C. Nee, M.W. Fu, J.Y.H. Fuh, K.S. Lee, and Y.F. Zhang. Determination of optimal parting directions in plastic injection mold design. *CIRP Annals*, 46(1):429–432, 1997.

[51] Julian Panetta, Qingnan Zhou, Luigi Malomo, Nico Pietroni, Paolo Cignoni, and Denis Zorin. Elastic textures for additive fabrication. *ACM Transactions on Graphics*, 34(4):135:1–135:12, jul 2015.

[52] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make it stand: balancing shapes for 3d fabrication. *ACM Transactions on Graphics*, 32(4):1, jul 2013.

[53] Alok K. Priyadarshi and Satyandra K. Gupta. Geometric algorithms for automated design of multi-piece permanent molds. *Computer-Aided Design*, 36(3):241–260, mar 2004.

[54] B. Ravi and M.N. Srinivasan. Decision criteria for computer-aided parting surface design. *Computer-Aided Design*, 22(1):11–18, jan 1990.

[55] Ryan Schmidt and Nobuyuki Umetani. Branching support structures for 3d printing. In *ACM SIGGRAPH 2014 Studio on - SIGGRAPH '14*. ACM Press, 2014.

[56] Christian Schumacher, Bernd Bickel, Jan Rys, Steve Marschner, Chiara Daraio, and Markus Gross. Microstructures to control elasticity in 3d printing. *ACM Transactions on Graphics*, 34(4):136:1–136:13, jul 2015.

[57] Adrian Secord, Jingwan Lu, Adam Finkelstein, Manish Singh, and Andrew Nealen. Perceptual models of viewpoint preference. *ACM Transactions on Graphics*, 30(5):1–12, oct 2011.

[58] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259, jan 2008.

[59] Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus Gross. Computational design of actuated deformable characters. *ACM Transactions on Graphics*, 32(4):1, jul 2013.

[60] Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. Recursive interlocking puzzles. *ACM Transactions on Graphics*, 31(6):1, nov 2012.

[61] Peng Song, Chi-Wing Fu, Yueming Jin, Hongfei Xu, Ligang Liu, Pheng-Ann Heng, and Daniel Cohen-Or. Reconfigurable interlocking furniture. *ACM Transactions on Graphics*, 36(6):1–14, nov 2017.

[62] Olga Sorkine and Marc Alexa. As-rigid-as-possible Surface Modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 109–116, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[63] Oded Stein, Alec Jacobson, and Eitan Grinspun. Interactive design of castable shapes using two-piece rigid molds. *Computers & Graphics*, 80:51–62, may 2019.

[64] Denis Sumin, Tim Weyrich, Tobias Rittig, Vahid Babaei, Thomas Nindel, Alexander Wilkie, Piotr Didyk, Bernd Bickel, Jaroslav Křivánek, and Karol Myszkowski. Geometry-aware scattering compensation for 3d printing. *ACM Transactions on Graphics*, 38(4):1–14, jul 2019.

[65] Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. Pteromys: interactive design and optimization of free-formed free-flight model airplanes. *ACM Transactions on Graphics*, 33(4):1–10, jul 2014.

[66] Nobuyuki Umetani, Athina Panotopoulou, Ryan Schmidt, and Emily Whiting. Printone: interactive resonance simulation for free-form print-wind instrument design. *ACM Transactions on Graphics*, 35(6):1–14, nov 2016.

[67] J. Vanek, J. A. G. Galicia, and B. Benes. Clever support: Efficient support structure generation for digital fabrication. *Computer Graphics Forum*, 33(5):117–125, aug 2014.

[68] J. Vanek, J. A. Garcia Galicia, B. Benes, R. Měch, N. Carr, O. Stava, and G. S. Miller. PackMerger: A 3d print volume optimizer. *Computer Graphics Forum*, 33(6):322–332, may 2014.

[69] L. Wang and E. Whiting. Buoyancy optimization for computational fabrication. *Computer Graphics Forum*, 35(2):49–58, may 2016.

[70] W. M. Wang, C. Zanni, and L. Kobbelt. Improved surface quality in 3d printing by optimizing the printing direction. *Computer Graphics Forum*, 35(2):59–70, may 2016.

[71] Weiming Wang, Haiyuan Chao, Jing Tong, Zhouwang Yang, Xin Tong, Hang Li, Xiuping Liu, and Ligang Liu. Saliency-preserving slicing optimization for effective 3d printing. *Computer Graphics Forum*, 34(6):148–160, jan 2015.

[72] Weiming Wang, Tuanfeng Y. Wang, Zhouwang Yang, Ligang Liu, Xin Tong, Weihua Tong, Jiansong Deng, Falai Chen, and Xiuping Liu. Cost-effective printing of 3d objects with skin-frame structures. *ACM Transactions on Graphics*, 32(6):1–10, nov 2013.

[73] Miaojun Yao, Zhili Chen, Linjie Luo, Rui Wang, and Huamin Wang. Level-set-based partitioning and packing optimization of a printable model. *ACM Transactions on Graphics*, 34(6):1–11, oct 2015.

[74] Chunjie Zhang, Xionghui Zhou, and Congxin Li. Feature extraction from freeform molded parts for moldability analysis. *The International Journal of Advanced Manufacturing Technology*, 48(1-4):273–282, sep 2009.

[75] Ran Zhang, Thomas Auzinger, Duygu Ceylan, Wilmot Li, and Bernd Bickel. Functionality-aware retargeting of mechanisms to 3d shapes. *ACM Transactions on Graphics*, 36(4):1–13, jul 2017.

[76] Xiaoting Zhang, Xinyi Le, Athina Panotopoulou, Emily Whiting, and Charlie C. L. Wang. Perceptual models of preference in 3d printing direction. *ACM Transactions on Graphics*, 34(6):1–12, oct 2015.