Controllable shape synthesis for digital fabrication

Jérémie Dumas

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Synthèse de formes contrôlable pour la fabrication digitale

THÈSE

présentée et soutenue publiquement le 3 Février 2017

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Jérémie Dumas

Composition du jury

Rapporteurs :	Paolo Cignoni	Directeur de recherche, ISTI-CNR, Italie
	Rüdiger Westermann	Professeur, TUM, Allemagne
Examinateurs :	Grégoire Allaire	Professeur, Ecole Polytechnique, France
	Marie-Paule Cani	Professeure, Université de Grenoble, France
	Isabelle Debled-Rennesson	Professeure, Université de Lorraine, France
Directeur de thèse :	Sylvain Lefebvre	Directeur de recherche, INRIA Nancy, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

"An idea a day keeps the boredom away."

Anonymous Researcher

ÉCOLE DOCTORALE IAEM LORRAINE

Abstract

Doctorat de l'Université de Lorraine

Controllable Shape Synthesis for Digital Fabrication

by Jérémie Dumas

The main goal of this thesis is to propose methods to synthesize shapes in a controllable manner, with the purpose of being fabricated. As 3D printers grow more accessible than ever, modeling software must now take into account fabrication constraints posed by additive manufacturing technologies. Consequently, efficient algorithms need to be devised to model the complex shapes that can be created through 3D printing. We develop algorithms for by-example shape synthesis that consider the physical behavior of the structure to fabricate. All the contributions of this thesis focus on the problem of generating complex shapes that follow geometric constraints and structural objectives.

In a first time, we focus on dealing with fabrication constraints, and propose a method for synthesizing efficient support structures that are well-suited for filament printers. In a second time, we take into account appearance control, and develop new by-example synthesis methods that mixes in a meaningful manner criteria on the appearance of the synthesized shapes, and constraints on their mechanical behavior. Finally, we present a highly scalable method to control the elastic properties of printed structures. We draw inspiration from procedural texture synthesis methods, and propose an efficient algorithm to synthesize printable microstructures with controlled elastic properties.

Keywords: 3D Printing, Fabrication Constraints, Shape Synthesis, Modeling, By-Example Synthesis, Procedural Texturing, Topology Optimization.

ÉCOLE DOCTORALE IAEM LORRAINE

Résumé

Doctorat de l'Université de Lorraine

Synthèse de formes contrôlable pour la fabrication digitale

par Jérémie Dumas

L'objet principal de cette thèse est de proposer des méthodes pour la synthèse de formes qui soient contrôlables et permettent d'imprimer les résultats obtenus. Les imprimantes 3D étant désormais plus faciles d'accès que jamais, les logiciels de modélisation doivent maintenant prendre en compte les contraintes de fabrication imposées par les technologies de fabrication additives. En conséquence, des algorithmes efficaces doivent être développés afin de modéliser les formes complexes qui peuvent être créées par impression 3D. Nous développons des algorithmes pour la synthèse de formes par l'exemple qui prennent en compte le comportement mécanique des structures devant être fabriquées. Toutes les contributions de cette thèse s'intéressent au problème de génération de formes complexes sous contraintes géométriques et objectifs structurels.

Dans un premier temps, nous nous intéressons à la gestion des contraintes de fabrication, et proposons une méthode pour synthétiser des structures de support efficaces qui sont bien adaptées aux imprimantes à filament. Dans un deuxième temps, nous prenons en compte le contrôle de l'apparence, et développons de nouvelles méthodes pour la synthèse par l'exemple qui mélangent astucieusement des critères sur visuels, et des contraintes sur le comportement mécanique des objets. Pour finir, nous présentons une méthode passant bien à l'échelle, afin de contrôler les propriétés élastiques des structures imprimées. Nous nous inspirons des méthodes de synthèse de texture procédurales, et proposons un algorithme efficace pour synthétiser des microstructures imprimables et contrôler leurs propriétés élastiques.

Mots-clefs : Impression 3D, Contraintes de fabrication, Synthèse de formes, Modélisation, Synthèse par l'exemple, Texturation procédural, Optimisation topologique.

Acknowledgments

On pourrait penser qu'il est facile d'écrire une ou deux pages de remerciements après avoir rédigé un manuscrit de plusieurs centaines de pages. Et pourtant, il est tellement facile d'oublier quelqu'un dans la liste qui suit... J'espère néanmoins ne léser personne dans cet exercice.

Mes premiers remerciements vont tout d'abord à directeur de thèse Sylvain Lefebvre, pour m'avoir encadré durant ces ≈ 3 années. Sa bonne humeur intarissable, sa patience à toute épreuve, et ses idées débordantes m'ont, je le pense, permis de m'épanouir dans des conditions de travail exceptionnelles. Je remercie également tous les membres de l'équipe ALICE, avec qui j'ai eu le plaisir de travailler : notre chef d'équipe, Bruno Lévy, à l'enthousiasme débordant qui donne du cœur à l'ouvrage ; les autres permanents de l'équipe, Dmitry Sokolov, Nicolas Ray, Samuel Hornus, Dobrina Boltcheva, Laurent Alonso, Rhaleb Zayer... mais également tous les membres temporaires (postdocs, ingénieurs, stagiaires), trop nombreux pour tous les citer sans en oublier, et avec qui j'ai pu échanger durant cette période.

Je remercie également mes co-bureaux successifs pour m'avoir supporté à longueur de journée : Jean Hergel, qui a partagé avec moi les péripéties de la vie de thésard ; Jonàs Martínez, pour les collaborations fructueuses ; et enfin Cédric Zanni, dernier arrivé dans l'équipe OpenGL. Merci également à Frédéric Claux pour son zèle, et à Maxence Reberol pour son pragmatisme. Enfin, merci également à tous ceux avec j'ai pu échanger virtuellement pour leur support. Je pense essentiellement à Emmanuel Jeandel, et à tous les collègues de `#sos` sur IRC.

Enfin, mes pensées vont aussi vers ma famille et mes amis, pour m'avoir permis d'obtenir un juste équilibre entre travail et loisirs. Je pense notamment à mes collègues de la danse, rencontrés au SUAPS, via l'association 3et4, en bals folk, ou à d'autres occasions sur Nancy ou même Strasbourg. Merci particulièrement à Marie-George, notre prof de danse au SUAPS, pour la bonne ambiance qui règne pendant ses cours. Enfin, je remercie également les autres doctorants qui j'ai pu rencontrer et fréquenter à l'occasion des séminaires organisés par l'Université de Lorraine. Ces événements sont particulièrement appréciables, dans un contexte qui favorise facilement l'isolement des doctorants dans leurs laboratoires respectifs.

For the English speaker who has read through my acknowledgments in French, many thanks for bearing with me. I would now like to express my gratitude towards the foreign colleagues I have interacted with during my PhD.

In particular, I would like to thank Li-Yi Wei for putting up with me during all this time. His unique style has taught me a lot, both in terms of organization, methodology and thinking. I need to thank also my co-workers, co-authors, or simply research colleagues, with whom I have interacted repeatedly over the years: Jun Wu, Haichuan Song, Qingnan Zhou, and many others.

I was fortunate enough to travel on multiple occasions during my PhD, and I was happy to visit our friendly colleagues at HKU, and attend a summer course on topology optimization organized by the formidable team of people at DTU.

Finally, I would like to thank the committee members of this thesis for taking the time to read and evaluate it, and for the interesting discussions that happened during the defense.

Contents

List of Abbreviations

CAD	**C**omputer **A**ided **D**esign
CNC	**C**omputer **N**umeric **C**ontrol
FDM	**F**used **D**eposition **M**odeling
FFF	**F**used **F**ilament **F**abrication
SLS	**S**elective **L**aser **S**intering
SLA	**S**tereo**L**ithography **A**pparatus
STL	**ST**ereo**L**ithography
DLP	**D**igital **L**ight **P**rocessing
CSG	**C**onstructive **S**olid **G**eometry
LDI	**L**ayered **D**epth **I**mages
LDNI	**L**ayered **D**epth **N**ormal **I**mages
FGM	**F**unctionally **G**raded **M**aterial
RVE	**R**epresentative **V**olume **E**lement
FEM	**F**inite **E**lement **M**ethod
SIMP	**S**olid **I**sotropic **M**aterial with **P**enalization
LSM	**L**evel **S**et **M**ethod
LSF	**L**evel **S**et **F**unction
RBF	**R**adial **B**asis **F**unction
OC	**O**ptimality **C**riteria
MMA	**M**ethod of **M**oving **A**symptotes
GCMMA	**G**lobally **C**onvergent **M**ethod of **M**oving **A**symptotes
CCSA	**C**onservative **C**onvex **S**eparable **A**pproximation
SAND	**S**imultaneous **AN**alysis and **D**esign
PCG	**P**reconditioned **C**onjugate **G**radient
GMG	**G**eometric **M**ulti-**G**rid
AMG	**A**lgebraic **M**ulti-**G**rid
PDE	**P**artial **D**ifferential **E**quation

To friends and family. . .

Résumé long

Introduction

L'objet principal de cette thèse est de proposer des méthodes pour la synthèse de formes qui soient contrôlables et permettent d'imprimer les résultats obtenus. Les imprimantes 3D étant désormais plus faciles d'accès que jamais, les logiciels de modélisation doivent maintenant prendre en compte les contraintes de fabrication imposées par les technologies de fabrication additives. En conséquence, des algorithmes efficaces doivent être développés afin de modéliser les formes complexes qui peuvent être créées par impression 3D.

Nous développons des algorithmes pour la synthèse de formes par l'exemple qui prennent en compte le comportement mécanique des structures à fabriquer. Toutes les contributions de cette thèse s'intéressent au problème de la génération de formes complexes sous contraintes géométriques et objectifs structurels.

Dans un premier temps, nous présentons au Chapitre 2 les différents travaux en rapport avec les thèmes développés dans cette thèse. Notamment, sont présentées : les différentes technologies de fabrication additive, les techniques de modélisation d'objets autour de l'impression 3D, les méthodes de synthèse de contenu par l'exemple (image ou modèles 3D), la modélisation de structures de remplissages et de microstructures pour l'impression 3D et enfin les méthodes de conception optimale de structures fort répandues dans le domaine de l'ingénierie mécanique.

Synthèse de formes et contraintes de fabrication

Au Chapitre 3, nous nous intéressons à la gestion des contraintes de fabrication et proposons une méthode pour synthétiser des structures de support efficaces qui sont bien adaptées aux imprimantes à filament.

La Fabrication par Fil Fondu (FFF) désigne le procédé de fabrication d'objets 3D à partir de filaments de plastiques fondus. Le plastique chaud sort de la buse et fusionne avec le morceau directement en dessous, ajoutant une couche de matière à l'objet en train de s'imprimer. Cependant, le filament peut être déposé uniquement *par dessus* une surface existante. De fait, les surplombs nécessitent d'être imprimés avec une *structure de support* jetable, qui vient supporter temporairement les fils de plastique, qui autrement se mettraient à pendre dans le vide.

Les techniques existantes pour la génération de supports se classent en deux catégories : la première permet de réaliser des impressions de manière extrêmement fiable en remplissant le dessous d'un objet par une structure dense, aux dépens d'une augmentation de la quantité de matière utilisée et du temps d'impression. La seconde catégorie génère une fine structure hiérarchique se connectant à la surface en un nombre limité de points. Cela gâche moins de matière, aux dépens de la fiabilité : l'objet peut devenir instable, la structure elle-même peut être difficile à imprimer et la qualité de surface sous l'objet est dégradée. L'utilisateur doit alors corriger la structure et ses paramètres pour chaque nouveau modèle à imprimer.

Nous proposons d'exploiter la capacité des imprimantes FFF à imprimer des ponts au dessus du vide. Un pont étant toujours supporté à ses deux extrémités par des piliers, ils sont à la fois plus résistants et plus stables qu'une structure d'arbre hiérarchique. Notre technique commence par sélectionner les points à supporter en fonction des surplombs et de la stabilité des pièces au cours du processus d'impression. Elle optimise ensuite une structure d'échafaudage imprimable qui comprend à la fois des ponts horizontaux et des piliers verticaux, afin de supporter les points nécessaires. Le résultat est une technique de génération de support automatique utilisant peu de matière, tout en garantissant une bonne qualité de surface et stabilité durant le processus d'impression.

Synthèse de formes et contraintes sur l'apparence

Au Chapitre 4, nous prenons en compte le contrôle de l'apparence et développons de nouvelles méthodes pour la synthèse par l'exemple qui mélangent astucieusement des critères visuels et des contraintes sur le comportement mécanique des objets.

Il existe plusieurs techniques pour synthétiser automatiquement des images 2D ressemblant à une texture exemple donnée en entrée. La plupart de ces approches consiste à optimiser une nouvelle image afin que les voisinages des couleurs dans la sortie correspondent à ceux de l'entrée, au travers des différentes échelles. Dans la Section 4.1, nous revisitons la synthèse de texture par l'exemple dans le cadre de la fabrication additive. Notre but est de générer non seulement des couleurs, mais également une structure le long d'une surface de sortie : étant donné un exemple indiquant les pixels *solides* et les pixels *vides*, nous générons un motif similaire le long de la surface de sortie. La difficulté principale est de garantir que le motif produit est non seulement formé d'une seule composante connexe, mais également qu'il est suffisamment solide d'un point de vue structurel.

Pour parvenir à cette fin, nous proposons une nouvelle formulation de synthèse de texture sur surface à partir d'exemple, qui fonctionne directement sur une fine couche de voxels autour de la surface. Il est alors possible de mettre à jour le motif localement et efficacement, ce qui permet à notre optimiseur structurel d'effectuer des changements qui améliorent la rigidité globale du motif. Nous utilisons cette technique dans un schéma itératif qui optimise alternativement l'*apparence* et la

solidité de la structure. Nous prenons en compte les contraintes de fabrication ainsi qu'une description fournie par l'utilisateur des forces extérieures auxquelles le modèle devra résister.

Nos résultats exploitent pleinement les possibilités de la fabrication additive en permettant aux utilisateurs de créer des formes complexes le long de surfaces. Ces structures sont sophistiquées, mais restent semblables aux exemples fournis en entrée, fournissant un outil de modélisation accessible à un utilisateur occasionnel.

Dans la Section 4.2, nous nous intéressons plus en détail au problème de combiner l'optimisation topologique et la synthèse de texture. Le domaine de l'optimisation topologique s'intéresse à la synthèse de formes avec objectifs structurels, comme par exemple obtenir la forme la plus rigide possible étant donné une certaine quantité de matière. Au-delà de l'étude des structures optimales, ces méthodes constituent des outils de conception de plus en plus populaires, car elles produisent automatiquement des structures possédant des propriétés physiques attrayantes, une tâche difficile à réaliser à la main même pour des dessinateurs expérimentés. Cependant, il n'existe pas de manière simple de contrôler l'apparence des objets ainsi générés.

Nous proposons d'optimiser des formes en considérant *à la fois* leurs propriétés structurelles *et* leur apparence, cette dernière étant contrôlée par un motif d'exemple fourni par l'utilisateur. Ces deux objectifs sont difficiles à combiner, car le caractère optimal d'une structure définit entièrement sa forme, ne laissant aucun degré de liberté pour contrôler l'apparence. Nous proposons une nouvelle formulation, où l'apparence est optimisée comme objectif, tandis que les propriétés structurelles sont considérées comme des *contraintes*. Cela produit des structures avec une rigidité suffisante tout en laissant une marge de manœuvre suffisante pour que l'apparence de la structure finale ressemble au motif donné en entrée.

Notre approche génère des formes rigides en utilisant une quantité de matière spécifiée, tout en respectant des contraintes optionnelles comme des trous, des zones solides, des points d'attache et des forces externes appliquées au système. L'apparence est définie par des images d'exemple, ce qui rend notre technique accessible à un utilisateur inexpérimenté. Nous illustrons l'utilisation de notre méthode dans le contexte de la fabrication en utilisant une découpeuse laser pour confectionner divers objets physiques à partir des formes optimisées automatiquement.

Dans la Section 4.3 traite du problème de l'optimisation jointe de la structure et de l'apparence sur des structures composées d'éléments *discrets*, dont les formes sont prédéfinies et qui constituent alors des agrégats de géométries.

Les agrégats de géométries composés de volumes répétitifs sont omniprésents dans la nature comme dans les objets fabriqués. Un important effort de recherche est mené afin de modéliser, calculer un rendu réaliste, ou animer des agrégats de géométries ayant une apparence visuelle et un comportement dynamique bien précis. Cependant, optimiser les propriétés mécaniques de ces agrégats à des fins

de fabrication tout en fournissant un contrôle intuitif sur leur apparence visuelle reste un problème difficile.

Nous présentons une méthode pour créer des agrégats de géométries qui sont à la fois mécaniquement solides pour la fabrication et visuellement fidèles à des exemples spécifiés par un utilisateur. Notre principale observation est que de tels agrégats contiennent souvent suffisamment de répétitions qui peuvent être optimisées afin de respecter à la fois les critères visuels et mécaniques sans nécessiter de supports additionnels. Nous visons à produire des résultats volumiques contenant à la fois des éléments discrets et continus.

Microstructures et contrôle des propriétés élastiques

Enfin, le Chapitre 5 présente une méthode passant bien à l'échelle, afin de contrôler les propriétés élastiques des structures imprimées. Nous nous inspirons des méthodes de synthèse de texture procédurales et proposons un algorithme efficace pour synthétiser des microstructures imprimables et contrôler leurs propriétés élastiques.

Les microstructures à l'échelle de dixièmes de microns affectent les propriétés physiques des objets, les rendant plus légers ou plus flexibles. Alors que ces structures sont difficiles à produire selon des procédés traditionnels, les techniques de fabrication additive nous permettent désormais de produire physiquement de telles microstructures à des coûts peu élevés.

Dans la Section 5.1, nous proposons d'étudier des microstructures procédurales et apériodiques inspirées de mousses à cellules ouvertes issues de diagrammes de Voronoi. L'absence de régularité permet, via une approche simple, de faire varier graduellement la géométrie de la mousse — et donc ses propriétés mécaniques — au niveau de la surface et à l'intérieur d'un objet cible. Plutôt que d'avoir recours à un procédé d'optimisation globale, les microstructures sont générées directement afin d'exhiber des propriétés élastiques bien précises. L'évaluation implicite est semblable aux textures procédurales en infographie et s'adapte localement afin de suivre le champ d'élasticité donné en entrée. Cela permet de générer des structures très détaillées dans des objets de grande taille sans avoir à produire une représentation explicite — maillage ou voxels — de la géométrie finale : les structures sont créées à la volée et chaque couche de l'objet peut être envoyée directement à l'imprimante.

Nous étudions le comportement élastique de ces microstructures et fournissons une description complète de la procédure pour les générer. Nous expliquons comment déterminer les paramètres géométriques de ces microstructures en fonction de l'élasticité désirée et évaluons le résultat sur des échantillons imprimés en 3D. Enfin, nous appliquons notre approche à la fabrication d'objets dont l'élasticité varie spatialement et décrivons un modèle implicite pour réaliser une armature le long de la surface de l'objet afin de connecter les microstructures intérieures sans transitions.

Chapter 1

Introduction

Computer graphics is the branch of computer science concerned with representing, modeling and visualizing the world — be it real or imaginary. In half a century, recent developments in computer graphics have considerably impacted animation and movie industries, video games, and interactive design, by producing ever more realistic content. While the discipline itself is fairly recent, earliest examples of computer graphics applications can be traced back to half a century ago, e.g. with the revolutionary *Sketchpad* by Sutherland [1964], which already showcased impressive features for its time. Nowadays, software and hardware capabilities have grown tremendously, to the point that a simple desktop computer is now able to render extremely complex scenes at interactive frame rates.

Up until recently, modeling software could be roughly classified in two categories: 3D computer graphics software, such as *Blender*, can be used for computer animation, visual effects, video games, while computer-aided design (CAD) software, such as *Catia* or *SolidWorks*, target more specifically applications in mechanical engineering or architectural design. While the former category of software provides accessible tools with a great deal of artistic control, they aim at creating *virtual objects*, and using them for creating physical structures poses a set of additional challenges. On the contrary, CAD software cater to specialists and "traditional" fabrication methods, but they remain difficult to use by non-experts for creating the complex objects now attainable through modern 3D printing technologies. Even for expert users, creating the shapes considered in this thesis can be extremely challenging via traditional approaches.

Most traditional manufacturing processes, used in industry, are subtractive: matter is carved out of a block of metal, wood, or any other material, by machining tools controlled by numerical commands, e.g. CNC milling machines. Typically, a rough tool is used to first carve the coarse outline of a shape, while a finer tool is used for the finishing operation. Before it can be manufactured, the shape is designed using a specialized CAD software. The parts can then be numerically simulated, and thoroughly studied and analyzed, before they can be put into production. This design phase is done by expert engineers with a good understanding of the mechanical properties of the object to be manufactured, and some expectations about the functionality it needs to perform. Even so, the design process is often

a painful iterative loop, where one has to simulate a piece, and update the model accordingly every time the results of the simulation does not match the expectations.

Over the past decade, 3D printing has grown tremendously, and presents itself as a viable alternative to traditional subtractive manufacturing methods. This rapid growth can be explained by the decline in hardware prices, and by the progress in software development which makes the technologies more accessible to end-users. While subtractive fabrication technologies remain the standard for *mass production* of industrial pieces, additive manufacturing has drawn a population of hobbyists and small entrepreneurs who seek to exploit its potential for *mass customization*.

Different from skilled engineers and professional designers, this new category of users need to be provided with appropriate software tools to help them modeling and fabricating complex objects, without any deep understanding of the actual fabrication process. Computer graphics has a long history of proposing modeling tools for non-technical users — e.g. CG artists, game modders, level designers — and these techniques hold great promises to help anyone model physical objects. However, they were never concerned with physical fabrication processes and manufacturing constraints. One example of such constraint is minimum thickness: one has to ensure the model being fabricated can 1) be printed correctly with the desired printing technology, and 2) the printed replica does not break under normal manipulation.

Thus, a central question we seek to answer in this thesis, is how to provide accessible design tools to model complex objects, and provide both artistic control, and satisfying fabrication requirements? This idea is illustrated Figure 1.1.

There are several approaches to providing artistic control in an application. In a large part of the work in this thesis, we draw inspiration from by-example texture synthesis methods, which seek to synthesize visually appealing content, whose appearance should resemble that of an input exemplar given by the user. By providing small patterns or elements which are easy to design by hand, an inexperienced user can quickly explore a range of new models which are generated automatically, from a small set of parameters. In fact, by-example synthesis methods help experts as well as non-experts, since designing these patterns is long and tedious. Texture synthesis techniques have been used successfully in games and in the movie industry, e.g. in the movie *Tangled* [Eisenacher et al. 2010]. In Chapter 4, we seek to develop new by-example synthesis methods that account for the physical restrictions imposed by a manufacturing process, in addition to the usual control provided by standard texturing methods.

A second aspect central to this thesis is the representation and manipulation of complex shapes. As manufacturing technologies grow, it is essential to devise algorithms that help us print objects which are ever more complex, at resolutions ever higher. In this document, Chapter 3 explains how to deal with certain fabrication constraints when printing complex shapes, while Chapter 5 presents a scalable

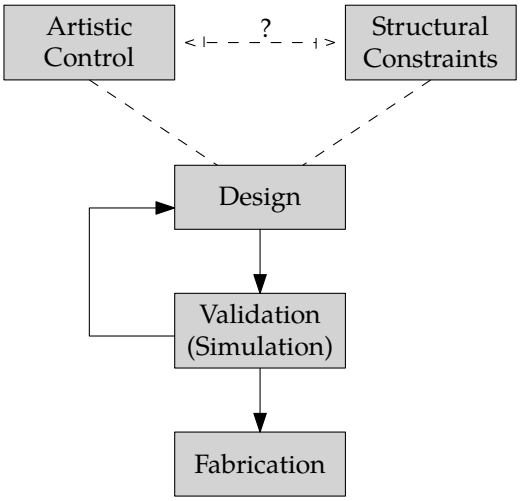


Figure 1.1 – A typical design pipeline for fabrication revolves around a central iterative loop: a model is first designed, then numerically simulated, until the desired constraints and objectives are met. A central question in this thesis is how to provide both artistic control and automated solutions that meet the constraints imposed by the target fabrication technology.

method to control the elastic properties of printed objects by varying their internal microstructure.

Additive Manufacturing. 3D printing technologies belong to the family of *additive manufacturing*, in opposition to the subtractive technologies mentioned earlier, such as CNC milling. Instead of carving matter from a block of material, the object is fabricated by depositing matter, usually in a layer-by-layer manner. Each layer is stacked on top of the precedent, by mean of different chemical processes, such as plastic being melted down for Fused Deposition Modeling (FDM), photosensitive resin being cured for Stereolithography (SLA), etc.

Compared to subtractive manufacturing, this has a number of advantages. The first and probably the most appealing one is the so-called "complexity paradox": the feasibility, material cost, and print time of a model no longer depends on how intricate the geometry is. It is now possible to fabricate complex objects that were not feasible before. Objects with more intricate geometries are often cheaper and faster to print with additive manufacturing techniques. The sparser the structure, the less expensive it is to 3D print. In a subtractive process, one has to consider the accessibility of a feature by the machining tool head, e.g. it is not possible to fabricate objects with cavities, even with a small aperture. Additive technologies also have certain restrictions, explained Section 2.1.1, but they are much less dramatic.

While it would appear that 3D printing is able to produce complexity for free, it is in fact an illusion: *the challenges are shifted to the software side*, which has to

deal with the model complexity, and *to the user*, who has to model the part in the first place. Printing larger and more complex objects, at very detailed resolutions, requires a careful software implementation, and ideally one should be able to *stream* information to the printer, so that only the necessary data is being processed at a given moment in time.

Last but not least, another important advantage of additive manufacturing, and probably what motivates the need for innovative design applications, is that they are tools of mass customization. Indeed, industrial manufacturing processes, e.g. injection molding, are oriented towards mass production of the same component over and over. This heightens the cost of fabricating two different objects successively, as one has to setup the equipment, recalibrate the machines, etc. With additive manufacturing, one can change the model anytime between two prints, without any cost. This makes 3D printing the method of choice for rapid prototyping applications, as a user can quickly explore different physical realizations of a design of interest.

Applications of 3D Printing. When the topic of 3D printing is brought up to unfamiliar users, the question of its potential applications is often raised. There seem to be some misconception among individuals outside the field that 3D printing is only used to create plastic toys. While the toy industry is in fact no small market, the reality is that additive manufacturing has applications in about just every aspect of our society, to the point that The Economist calls it the third industrial revolution[1]. To give some perspective on the matter, we present just a small percentage of potential applications of 3D printing.

Undoubtedly one of the most promising use of 3D printing is for medical applications. The ability to synthesize tissue to repair damaged organs, or to fabricate prosthetics — be it for tooth, or an artificial limb [link] — which are tailored to an individual at no additional cost, is pushing the limits of medical science. As the technology progresses, 3D printers are able to reproduce a wider range of structures, at higher and higher resolutions. A recent example for manufacturing bone structures with biomaterials is given in [Jakus et al. 2016]. From a software perspective, it means there is a necessity to develop efficient algorithms which can handle the increasing complexity of the underlying 3D models.

Another exciting field impacted by 3D printing is archaeology and the preservation of cultural heritage. Scanning technologies enables scientists to digitalize on object somewhere, and print it on the other side of the globe for another team to study it. Working with physical copies avoids damaging further the original object, which can be restored digitally for example. Furthermore, scanning and 3D printing archaeological sites allows humanity to preserve important landmarks, which can be destroyed, e.g. after a natural disaster, or in times of war [link].

[1] http://web.archive.org/web/20120515040242/http://www.economist.com/node/21552901

A related domain which benefits from these new rapid prototyping methods is architectural design. From the realization of miniature mock-ups, to the fabrication of actual real-life buildings such as a pavilion [link], 3D printing is providing new ways for architects and designers to design new structures. More generally, the construction industry may also rely on 3D printing, e.g. to provide a quick and easy way to install new shelters after an earthquake, at a very low cost. The technology can also be used to build fancier accommodation, such as a small castle [link].

In the automobile and aerospace industry, manufacturers are also starting to take advantage of the possibilities offered by additive manufacturing, for example to fabricate more functional aircraft brackets, e.g. using titanium [link]. See also [Tomlin and Meyer 2011] for the optimization of an aerospace part for metallic additive manufacturing. At a different scale, manufactured parts also find they way into robotic components. For example, the Poppy project features an open-source design of the different parts of a humanoid robot, which can be easily customized. 3D printed parts can also be extremely useful for home improvement, e.g. to print drill guides (thing:402531 and thing:267196), replacement parts, attachments, etc.

Arts and craftsmanship are other domains where 3D printing brings interesting developments. Most notably, resin printers — which have a very high-precision, but a limited build volume — are very popular for jewelry [link]. The 3D printer allows a creator to quickly experiment with different designs and physically appreciate their quality. Once the creator is satisfied, the printed model is used to create a mold, which is used to cast one or more copies of the final object. 3D printing can also be used to create complex sculptures, such as this beautiful 3D printed zoetrope.

Finally, there are still a number of applications that we did not evoke: garment design and clothing, such as the products proposed by the company *Nervous System* [Rosenkrantz and Louis-Rosenberg 2007] ; rigging and character animation [Glauser et al. 2016] ; 3D printed food [link]. More applications of 3D printings can be found everyday on specialized news webzine, such as https://all3dp.com/ or https://3dprintingindustry.com/.

Contributions. In light of the preceding discussion, the contributions of this thesis can be described as follows. In Chapter 3, we develop new algorithms for handling fabrication constraints in complex virtual objects. In particular, we propose a technique for synthesizing support structures that are very reliable, but also efficient in terms material costs and print time. External support structures are typically used to print shapes that are not attainable on regular filament printers due to manufacturing constraints (overhangs).

In Chapter 4, we explore several tools for by-example synthesis of structured content that considers the fabrication process. While standard by-example synthesis methods do not optimize the mechanical behavior of the final models, we present several algorithms for automatically synthesizing content from a small exemplar pattern so that *structurally sound* objects are produced. With a few *concurrent* works

exploring similar ideas, we were one of the first to propose combining by-example artistic control and fabrication requirements.

In Chapter 5, we draw inspiration from procedural modeling techniques, and develop an efficient method for synthesizing microstructures inside a model. The algorithm produces spatially-varying microstructure with prescribed elastic properties, and the resulting geometry can be sent directly to the printer in an *online* manner.

The work presented in this thesis is the result of fruitful collaborations with various talented colleagues, and lead to the following publications at international venues:

- Dumas, J.; Hergel, J. and Lefebvre, S. [2014]. "Bridging the Gap: Automated Steady Scaffoldings for 3D Printing". *ACM Trans. Graph.* 33.4, 98:1–98:10. DOI: 10.1145/2601097.2601153
- Dumas, J.; Lu, A.; Lefebvre, S.; Wu, J. and Dick, C. [2015]. "By-Example Synthesis of Structurally Sound Patterns". *ACM Trans. Graph.* 34.4, 137:1–137:12. DOI: 10.1145/2766984
- Martínez, J.; Dumas, J.; Lefebvre, S. and Wei, L. [2015a]. "Structure and Appearance Optimization for Controllable Shape Design". *ACM Trans. Graph.* 34.6, 229:1–229:11. DOI: 10.1145/2816795.2818101
- Hornus, S.; Lefebvre, S.; Dumas, J. and Claux, F. [2016]. "Tight Printable Enclosures and Support Structures for Additive Manufacturing". *Eurographics Workshop on Graphics for Digital Fabrication*. The Eurographics Association. DOI: 10.2312/gdf.20161074
- Martínez, J.; Dumas, J. and Lefebvre, S. [2016]. "Procedural Voronoi Foams for Additive Manufacturing". *ACM Trans. Graph.* 35.4. DOI: 10.1145/2897824.2925922

Chapter 2

Related Work

Digital fabrication is an inherently cross-disciplinary topic, with applications in design, arts, geometry, physics, mechanical engineering, robotics, electronics, and many more. In this chapter, we present the different fundamental concepts used throughout this thesis. Section 2.1 introduces the different digital fabrication technologies used in this thesis, Sections 2.2 and 2.3 focuses on the shape synthesis, in the context of fabrication and via by-example methods, while Sections 2.4 and 2.5 are more concerned with the mechanical engineering aspects of synthesizing micro- and macro- structures respectively. Each section attempts to be a self-contained introduction covering the most relevant literature in the field, and to give an overview of the associated past and concurrent related work.

In Section 2.1, we describe the different manufacturing technologies considered in this work, and the constraints associated to them. We also briefly explain the slicing process and the toolpath planning, whose goal is to convert a digital 3D model in a set of machine instructions that command the printers. Finally, we review existing approaches for enforcing geometric constraints on a model to make it manufacturable.

In Section 2.2, we discuss more generally how to design shapes for digital fabrication. Modeling tools need to be adapted to produce physical and tangible objects rather than purely virtual content. A new class of problems arises from additive manufacturing constraints, such as structural analysis, mass distribution, smart design editing or optimization. In this section, a particular attention is given to modeling tools that attempt to help users in their design choices, be it through interactive user interfaces, or via an automatic optimization procedure .

In Section 2.3, we cover specifically question of by-example content generation, such as by-example texture synthesis, which is a long standing and fundamental problem in computer graphics. By-example synthesis methods seek to replicate the appearance of an input pattern or model, to generate content in a larger domain — e.g. an image in 2D, a surface or a volume in 3D. Appearance-based and by-example synthesis play a central role later in the Chapter 4 of this thesis.

In Section 2.4, we present in more details problems related to the generation of internal structures and microstructural patterns inside a target volume. Classical

infill generation aims to save up material by reducing the amount of matter printed inside an object, but other objectives such as print time and discretization errors can be taken into account. At a finer scale, changing the microstructure geometry of the printed volumes can also affect their macro-scale mechanical behavior. This effect is captured by what is known as the homogenization theory, which constitute an important part of the mechanical engineering literature. Efficient microstructure generation via procedural methods is also discussed. Meta-material design via the control of microstructures is the object of Chapter 5.

Finally, in Section 2.5, we discuss topology optimization, a discipline that belongs to the field of mechanical engineering. In particular, we review in more details the SIMP approach — one of the most popular form of topology optimization — as it was used extensively during the course of this work, especially in Sections 4.2 and 4.3. Furthermore, a particular attention will be given to the aspects of topology optimization most relevant to our work. First, we present methods that combine topology optimization with additive manufacturing constraints, such as overhangs or thickness control. Second, we examine a certain type of topology optimization methods, which considers what we call *discrete elements*, that can move in the domain as part of the optimization procedure. Synthesis methods that rely of those discrete elements the focus of Section 4.3.

2.1 Additive Manufacturing Processes

The rapid development of additive manufacturing technologies over the past decade has fostered a lot of interest from researchers in many different fields. In this section, we do not aim to present a comprehensive overview of all the possible manufacturing technologies, which would be beyond the scope of this thesis. Instead, we will focus on the technologies we used in our lab, as they are they are the ones I have worked with more extensively: filament printers, powder-based inkjet printers, laser-cutters, and resin printers. For a presentation of broader aspects of digital fabrication, the reader is referred to several of the recent surveys in the domain. [Schmidt and Ratto 2013] present challenges pertaining to the design of software that are tailored for additive manufacturing applications. [Oropallo and Piegl 2015] offers a synthetic presentation of ten challenges related to the 3D printing work flow. Guessasma et al. [2015] discuss optimization problems related to additive manufacturing. Finally, [Gao et al. 2015b] review different printing technologies and discusses problems related to the fabrication pipeline.

The rest of the section is organized as follows. In Section 2.1.1, we first present the manufacturing technologies used throughout the rest of this work, as well as the constraints they impose on the shapes to fabricate. In Section 2.1.2, we discuss the slicing pipeline, i.e. how to transform a virtual model into a set of instructions for the machine. In particular, we examine how image-based representations of solid objects can help simplifying this process. Finally, in Section 2.1.3, we review

established solutions that exist to enforce the aforementioned fabrication constraints, such as overhangs and minimum thickness. Note that methods which consider the mechanical behavior and structural soundness of printed objects, e.g. for analysis or editing purposes, are discussed later in Section 2.2.

2.1.1 Printing Technologies and Constraints

There are many fabrication processes that falls under the nomenclature of *additive manufacturing*. Their common characteristic is that they fabricate on object by adding matter, usually in a layer-by-layer manner, until the final shape is formed. This is in opposition with subtractive technologies, which operate by removing matter from a bloc of base material, e.g. CNC milling — one of the most common fabrication process used in industry.

There are different ways to categorize additive manufacturing technologies. If one considers the way layers are represented, an approach common to computer graphics would be to distinguish between vector graphics representations, which rely on polygonal descriptions of the printing toolpath — as in FDM or SLS printers —, and raster representations, which rely on a pixel grid, or bitmap image — as in DLP printers. A second approach would be to consider the physical process involved in the matter solidification — e.g. filament deposition or light polymerization. We chose to employ this latter classification, as it makes it easier to tie the fabrication constraints with the underlying physical process.

While the technologies presented here are among the most widespread 3D printing techniques, they do not represent an exhaustive list. For a more comprehensive presentation of the different additive manufacturing technologies, the interested reader is referred to recent books such as [Gibson et al. 2014]. In the following, we describe the shape properties and the different manufacturing technologies useful for this work. A summary of the printing constraints according to each technology is presented Table 2.1, and the different printing constraint are illustrated in Figure 2.2.

Regularity. Most of the technologies presented in this section have a limit of the minimum thickness that can be realized on a print. The dual, the limit on the minimum size of a hole, is also present on some technologies. Although, at the scale we consider, it is sometimes not clear if the limit is due to the imprecision of the machine toolpath, or to the physical manufacturing process. Before describing the particularities of the different technologies, we give a more precise definition of shape regularity, and what we mean by minimum thickness and minimum hole size. The definitions given here are inspired by [Williams and Rossignac 2005].

Definition 2.1.1 (Stability)**.** Let S be a 3D shape. We define the *stability* of a point $p \in \mathbb{R}^3$ as the radius of the largest (open) ball containing p that is completely enclosed in S or its complement $\overline{S}$. (See Figure 2.1.)

Definition 2.1.2 (Inner-regular)**.** Let S be a 3D shape, and $r \in \mathbb{R}^+$. We say that S is *r-inner-regular* if all the points inside S have a stability $\geqslant r$.
Definition 2.1.3 (Outer-regular)**.** Let S be a 3D shape, and $r \in \mathbb{R}^+$. We say that S is *r-outer-regular* if all the points inside the complement $\overline{S}$ have a stability $\geqslant r$.

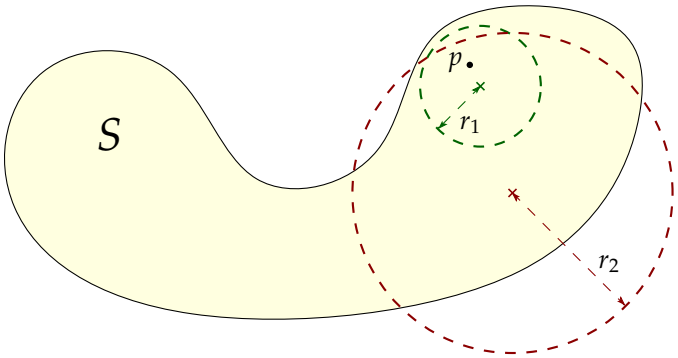


Figure 2.1 – Stability of a point p. The green disk contains p and is enclosed in the shape S, so p has a stability $\geqslant r_1$. Conversely, there is no open disk of radius r_2 that contains p and is enclosed in S or $\overline{S}$, so the stability of p is $< r_2$.

In the rest of this document, when we refer to the *minimum feature size*, or *minimum thickness*, that is achievable when printing a shape S, we mean that the shape S we send to the printer needs to be r_i-inner-regular, where r_i is the radius of the minimum printable feature. Note that the definition becomes more complex when considering machines such as filament printers, where the minimum feature size is not the same on XY plane than along the Z axis, so the balls would need to be defined according to a different metric. Similarly, when we refer to the *minimum hole size*, or *minimum void*, that can be achieved by a machine, we mean that the input shape S must be at least r_o-outer-regular, where r_o is the minimum hole size that can be realized by the printer.

Connectivity. A constraint that is common to all the technologies described in the following, is part connectivity. Unless the goal is to print non-assembly mechanisms, such a ball joints [Calì et al. 2012], it is undesirable to fabricate an object comprised of multiple connected components, as they would simply fall apart when the user manipulates the object. Note that connectivity is often not enough to guarantee that there are no structural defects in the final print. For example a large part connected to the rest of an object through a single tiny junction will be very fragile. To create more robust shapes, it is thus preferable to enforce a *sufficient* connectivity. This can be done at the geometrical level — by considering combinatorial terms such as the isoperimetric number of a graph in [Cignoni et al. 2014] —, or at a mechanical level — by simulating the physical behavior of the structure. Analysis of the structural soundness of printed shapes is discussed further in Section 2.2.1.

Extrusion Printers. Also called *filament printers*, this category encompasses the most popular and inexpensive consumer-level 3D printing technology, known under the equivalent terms of *Fused Deposition Modeling* (FDM) or *Fused Filament Fabrication* (FFF). A plastic filament is pulled by a motor through a heated print-head, which can move horizontally, melts down the plastics, and deposits it on a descending build plate. See Figure 2.3 for an illustration of the process. In our team we currently have a number FDM printers: *MakerBot Replicator 1*, *ORD Solutions RoVa3D 5 Extruder*, *Ultimaker 2*, *MakerBot Replicator 2*, and a homemade *Delta Robot* .

Because filament can only be deposited on top of another solid part, there is a limit to the negative slope that can be realized by a FDM printer Figure 2.2. This constraint on the minimum angle allowed on this negative slope is called the *overhang* constraint, and it depends on the layer height and nozzle width. A corollary of the overhang constraint is that *local minima* in the printing direction are to be avoided on the solid phase, otherwise the filament will simply fall onto the print bed. To print long overhang features, one typically resorts to a *support structure* built alongside the object, to prevent the filament from falling. This auxiliary structure is to be removed from the fabricated object during the print cleaning step.

FDM printers have also a *minimum thickness* limit to what they can print, determined horizontally by the nozzle width, and vertically by the minimum layer thickness the machine can achieve. On the other hand, there is no minimal hole constraints, and internal cavities are not a problem for FDM printers, contrary to other technologies presented in this section.

Note that despite the constraint on the slope angle, FDM printers have the remarkable ability to print horizontal lines, as long as they are supported at both extremities. These horizontal segments are called *bridges*, and will be heavily exploited in Section 3.1. The phenomenon is illustrated Figure 3.3, and discussed in more details in Section 3.1.2.

Powder Bed Printers. This second category of additive manufacturing technologies includes inkjet head printing, but also Selective Laser Sintering (SLS) and its variants. In our team we are currently using a *ZCorp 450*, which belongs to the family of inkjet 3D printers, sometimes called binder jetting. The common characteristic of powder bed printers is that they solidify a base material in a layer-by-layer manner. In the case of inkjet 3D printers, the print head moves horizontally and deposits a liquid binder material on the current layer, which has the effect of gluing together the powder particles. In the case of SLS printers, a laser is focused along the path where the layer is to be solidified. Note that inkjet printers have the advantage that they can inject standard inkjet colors into the powder, producing objects with a wide color range that is difficult to achieve with FDM printers.

Contrary to FDM printers, overhangs are not a problem with powder bed printers, but there is still a limit to the minimum thickness that can be achieved. In addition, there should not be any *enclosed cavities*, as the powder base material needs to be

extracted from the print. In this document, we will sometimes employ the equivalent terms *enclosed void*, or *pockets*. Note that, while overhangs are a non-issue on powder bed printers, it might still be necessary to print support structure alongside the target object. For example, if the solid volume is large and heavy, it will press down on the powder below, causing the whole print to sag. The can be avoiding by providing a small amount of supporting pillars during the build. With other technologies, such as metal laser sintering, the heat dissipation rate might be greatly improved with a support structure that diffuses the heat evenly throughout the volume.

Light Polymerization. Also called *resin printers*, this type of machine operates by solidifying a photo-sensitive polymer, which lies in a resin tank, in a layer-by-layer manner similar to powder bed printers. Contrary to powder bed printers, the model is often built upside-down, with the print platform *pulling* the object from the tank. On Stereolithography printers, the resin is cured via a laser, whose focus point is generally controlled through a set of mirrors. By contrast, DLP[1] printers use an image lit from a projector to solidify the resin at each layer. The printers used in our lab, a *B9 Creator V1.2* and a *Autodesk Ember*, are DLP-based resin printers.

Historically, stereolithography was also the first additive manufacturing technology to be invented. The first prototypes date back to the 1980s, first realized by Kodama [1981], and later patented simultaneously in France by André et al. [1984], and in the United States by Hull [1984]. Interestingly, the STL file format, that is widely used in computer graphics, CAD, and modeling software, stands for *stereolithography*. Although originally created for CAD systems, with the goal of producing parts through additive manufacturing on resin printers, STL is arguably not the best file format[2] for 3D printing applications nowadays.

In terms of constraints, resin printers can print slopes at almost any angles, so overhangs are usually not an issue. However, the object still needs to be attached to the print platform, otherwise it will "float around" in the resin tank. Consequently, one should still take care to avoid local minima in the print direction, which we also call *islands*. Similarly to powder bed printers, cavities are to be avoided, otherwise it would trap some amount of liquid resin inside. Finally, the minimum thickness requirement still persists, and there is also a minimum limit on the hole size, due to how the light diffuses in the solidification process. See [Jansen et al. 2013] for a discussion on this topic in the case of electron beam lithography, for the manufacturing of micro/nano-scale structures.

2D Cutting Machines. While not technically an additive manufacturing technology — 2D cutting machines operate by cutting matter *out* of a plank of base material —, cutting machines, and especially laser cutters, are becoming increasing accessible to the masses, and widely used for rapid prototyping purposes. Moreover, it is

[1] Digital Light Processing

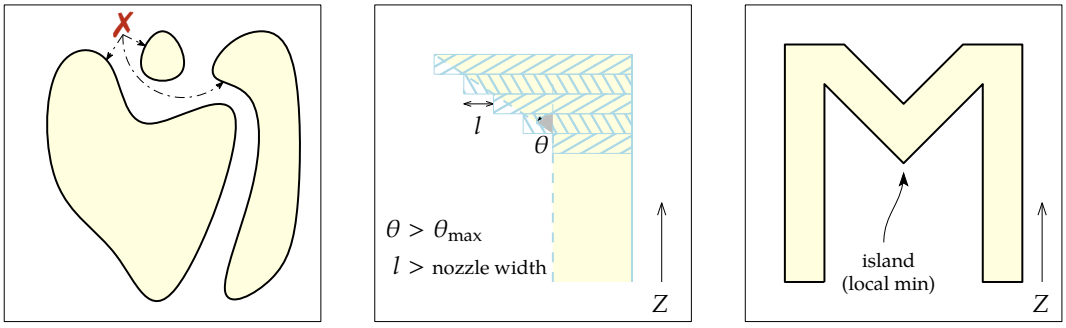
[2] https://medium.com/3d-printing-stories/why-stl-format-is-bad-fea9ecf5e45

possible to "stack" laser-cut pieces together, to assemble multiple layers into a 3D object. One can also "bend" laser-cut 2D sheets to create 3D shapes, as in [Mueller et al. 2013]. In the context of this thesis, laser-cut pieces were used extensively in Section 4.2 to fabricate 2D designs, and some were assembled and glued together to create 3D objects.

Since the designs produced by a laser cutter is restricted to 2D, there is no constraint such as overhangs or islands. Holes or cavities are usually not a problem, but it depends on the machines. For example, industrial water-jet cutters have some constraints on the type of holes they can cut. Finally, 2D cutting machines have also have a minimum achievable thickness and hole size, albeit this limit is usually much lower compared to the aforementioned 3D printing technologies.

Technology	Connectivity	Overhangs	Islands	Cavities	Minimum thickness	Minimum void
Filament	●	●	●	○	●	○
Powder	●	○	○	●	●	◐
Resin	●	○	●	●	●	●
Cutting (2D)	●	○	○	◐	●	●

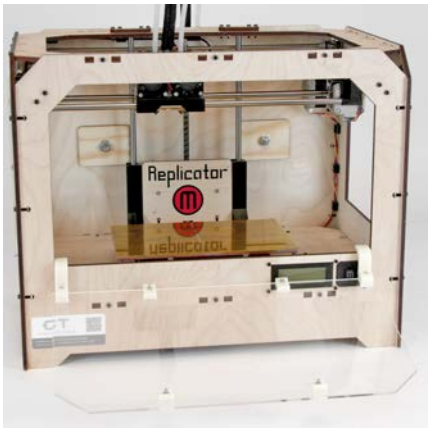
Table 2.1 – Summary of the different fabrication constraints depending on the printer category.

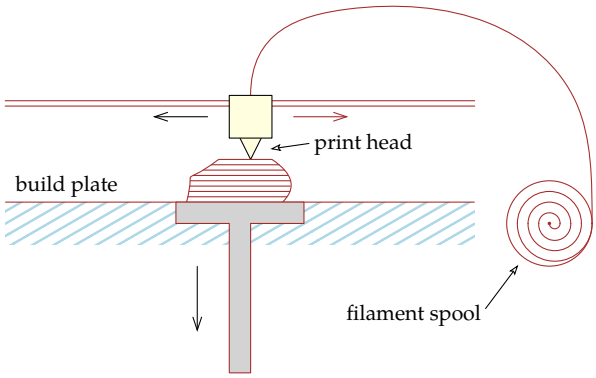


(a) Connectivity. **(b)** Overhangs. **(c)** Islands.

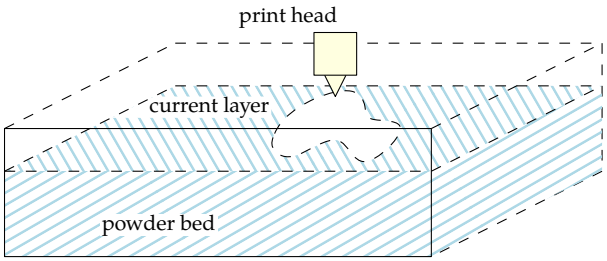
(d) Cavities. **(e)** Min thickness. **(f)** Min void.

Figure 2.2 – Geometric constraints in fabrication.

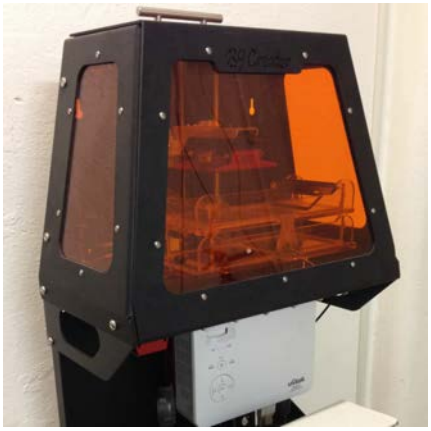


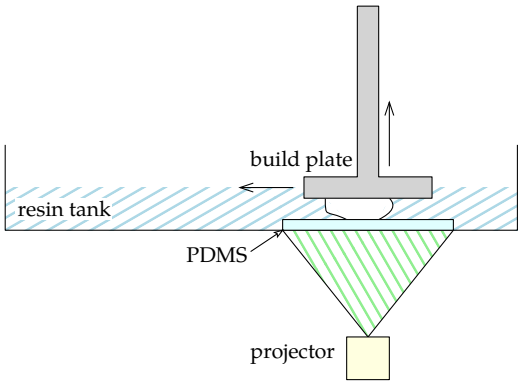


(a) Principle of a filament printer.
Image: https://www.flickr.com/photos/creative_tools/8266027093



(b) Principle of an inkjet powder bed printer.
Image: https://www.flickr.com/photos/neontommy/8264971129





(c) Principle of a DLP resin printer.
Image: https://www.flickr.com/photos/mangtronix/12331560733

Figure 2.3 – Illustration of 3D printers most used in this thesis.

2.1.2 Converting 3D Models to Printed Objects

Data Structures

Printer Input. Different printers use different data structures to represent the object that need to be fabricated. Filament printers typically need to be given the precise toolpath followed by the print heads — e.g. using G-code or `cli` format —, while powder bed and DLP resin printers require an image for every layer of the object to manufacture.

However, most digital objects are stored on the computer using a different representation. CAD systems typically use NURBS to describe piecewise polynomial surfaces modeling an object, while most consumer-level modeling applications preferably store a discrete representation in the form a triangle mesh $\mathcal{M} = (V, F)$, where $V \in \mathbb{R}^{n\times 3}$ denote the vertices of the mesh, and $F \in \mathbb{N}^{m\times 3}$ denote the facets of the mesh. Meshes can have quadrilateral or polygonal faces, and 3D models can also be represented by an implicit function or a fractal. Another possibility is to use volumetric data, either in the form of a dense voxel grid, a hierarchical grid (octree), or via a ray-based representation (dexels [Van Hook 1986], or layered depth images [Shade et al. 1998]).

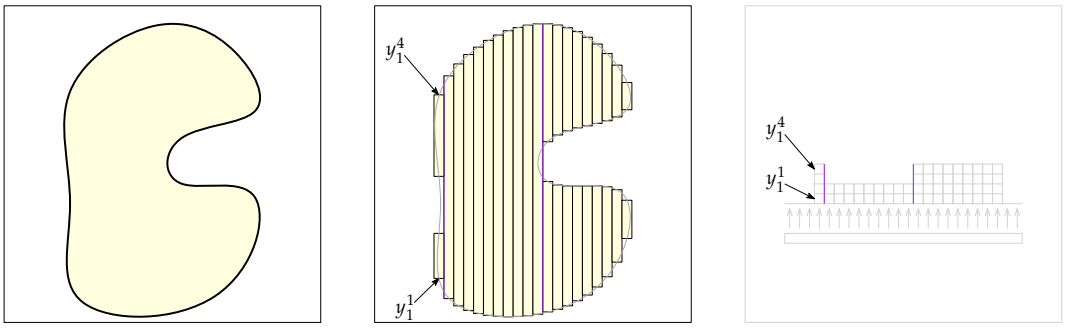
When the digital representation differs from the printer representation, a conversion needs to be done. This conversion is usually referred to as the *slicing* process. It also includes, in the context of filament printers, a step known as *path planning*.

Ray-Based Representations. By intersecting a regular grid of parallel rays with a 3D model, one can compute the segments from each ray that lie within the volume enclosed by the 3D model. Storing the endpoints of those segments yields a compact description of the original volumetric shape. The resulting discretization is called a ray-based representation, and it has important applications in rapid prototyping. Historically, the first ray-based solid representation was based on notion of dexels (for *depth pixels*), proposed by Van Hook in 1986. Van Hook [1986] proposed a technique to compute the results of CSG operations in image-space via this *dexel structure*, for the purpose of facilitating NC milling path-planning. A similar technique is now implemented in IceSL [Lefebvre 2013], a slicing software developed in our team. This data structure is illustrated Figure 2.4. Note that, while in the original paper by Van Hook [1986], each element in the linked-list stores a tuple $(z_{\min}, z_{\max})$ corresponding to the solid segment encoded by a dexel, in Figure 2.4 this representation has been "flattened out". In that aspect, a dexel buffer can be interpreted as a special case of a A-buffer, a technique developed for achieving order-independent transparency [Carpenter 1984; Maule et al. 2011].

Interestingly, *Layered Depth Images* (LDI) [Shade et al. 1998] describe a data structure similar to the dexel buffer, but were developed in a different context. The goal of LDI was to achieve efficient image-based rendering, while dexel buffers were used

for CSG operations, and A-buffers for rendering transparency. Consequently, the algorithms developed to build and render LDI are different than those involving dexel buffers, even though the underlying data structures are similar. In the context of digital fabrication, *Layered Depth Normal Images* (LDNI) — which are LDI augmented with surface normal information — have been proposed an alternative way to discretize 3D models [Huang et al. 2014a].

Ray-based representations are extremely appealing, as they allow to perform a number of operations directly in image-space, avoiding the for expensive remeshing techniques. This includes CSG operations, but also support requirement calculations, toolpath planning, infill calculations, etc. Implicit surfaces can also be discretized directly without prior explicit meshing. The drawback of ray-based data structures is that the discretization error is uniform across the volume, contrary to a triangle mesh which can use finer triangles around delicate features. However, since 3D printers have also a limited resolution, if one can provide a dexel buffer at the same resolution than the printing precision, then the space of 3D shapes that can be represented by a dexel buffer is in fact a superset of the actual shapes that can be fabricated.



(a) Input shape. **(b)** Dexels approximation. **(c)** Compact storage.

Figure 2.4 – The dexel-buffer data structure. An input shape (a) is approximated by its intersection with an axis-aligned grid of parallel rays (b), and stored compactly as an array or a linked-list (c).

Slicing Pipeline

Once the digital model representation, and the target 3D printing technology, have been chosen, one can proceed to convert the digital model into a set of machine instructions. In its simplest form, the slicing process amounts to computing the intersection between the input model and horizontal planes at each layers height. In a second step, specially for filament printers, one needs to transform each layer description — commonly a set of polygonal lines defining the slice data — into a sequence of paths to be followed by the print head. The output machine instructions are usually presented in the form of G-code, but it can also be a proprietary format, or a simple stack of images, as in the case of the *Autodesk Ember*.

The software carrying this transformation from the digital model to the machine instruction is called a *slicer*. There is a wide choice of slicing software available for filament printers. Popular choices include vendor-specific slicers, such as *Cura*[1] by *Ultimaker*, or *MakerWare* by *Makerbot*, or vendor neutral applications, such as *Slic3r*, *Repetier Host*, *CraftWare*, *MatterControl*, *Simplify3D*, among many others.

A detailed overview of the whole pipeline can be found in dedicated surveys, regarding slicing procedures [Pandey et al. 2003], or the process planning [Kulkarni et al. 2000]. More recently, the question of developing modeling and design tools specific to 3D printing applications has also been covered in SIGGRAPH and SIGGRAPH ASIA courses [Liu et al. 2014a; Umetani et al. 2015; Dinh et al. 2015].

Improving Surface Quality. As machines have limited resolutions, printed models suffer from artifacts inherent to the technology being used. This is typically known as aliasing in computer graphics. The printed surface quality can be improved by playing on the model orientation, adapting the slicing process, or segmenting the object into parts. Thrimurthulu et al. [2004] study the problem of finding the best model orientation to reduce build time and improve the surface finish on filament printers. More recently, Delfs et al. [2016] proposed another method to optimize the model orientation, in order to improve the surface roughness on parts printed on SLS machines.

Pintus et al. [2010] have proposed a shape enhancement technique to increase the amount of details perceived on a model to be printed. [Wang et al. 2015] present an adaptive slicing algorithm based on a visual metric, to reduce printing time while preserving salient features.

A third option is to partition the input shape to improve the printed surface quality. [Hu et al. 2014] decompose a shape into pyramidal parts so they can be fabricated on filament printers without supports. In a similar approach, Herholz et al. [2015] allows small surface deformations to reduce the number of parts in the resulting segmentation. Hildebrand et al. [2013] propose segmentation algorithm to reduce the approximation error between an input shape and the stack of slices resulting from the fabrication. In a recent work, Wang et al. [2016a] present a segmentation technique to improve surface quality, by relying on the fact that 3D printers generally have a higher resolution in the Z direction than along the XY directions. This echoes Answering different challenges, Schüller et al. [2014] proposed a method to fabricate bas-reliefs that depict certain target shapes viewed from specific directions.

Appearance and Colors. For 3D printers capable of fabricating colored objects, a number of approaches have been proposed to improve the quality of the resulting surfaces. Cignoni et al. [2008] and Brunton et al. [2015] proposed techniques to improve colored results on multimaterial printers, while Hergel and Lefebvre [2014]

[1] https://github.com/Ultimaker/CuraEngine

and Reiner et al. [2014] deal with the case of filament printers having multiple heads. There has also been an increasing interest in reproducing reflective properties in fabricated replica, such as materials with custom surface reflectance [Matusik et al. 2009; Weyrich et al. 2009], subsurface scattering [Hašan et al. 2010; Dong et al. 2010], 4D reflectance functions [Malzbender et al. 2012], translucent materials [Papas et al. 2013] or bi-scale materials [Lan et al. 2013]. Recently, other approaches to produce high-quality, faithful, colored printouts have been explored. Zhang et al. [2015e] and Panozzo et al. [2015] use water transfer printing to texture map an input image onto arbitrary surfaces. Schüller et al. [2016] use thermoforming to achieve similar goals, but achieve higher-quality results with an easier hardware setup.

Alternative Prototyping Schemes

While not directly the object of our work, alternative schemes for rapid prototyping can provide interesting views on fabrications constraints. The goal of these tools is to empower users with ways to quickly create physical replica of 3D models, using using alternative fabrication methods. Garg et al. [2014] use wire meshes, that can shear, but not bend, to replicate a user-given surface. Mueller et al. [2014a] presented a process to rapidly print a wireframe preview of a given 3D model. Skouras et al. [2015] proposed a design tool to create 3D models from interlocking elements, which can be cut from a piece of paper. More recently, interlocking structures using steel rods has also been investigated in [Miguel et al. 2016]. Another way of fabricating objects is by using LEGO bricks, e.g mixed with additive manufacturing [Mueller et al. 2014b], or directly to replicate large models with a minimum number of bricks, while maintaining a feasible solution [Luo et al. 2015]. Objects made of intersecting planar cross-sections is also a popular way to quickly fabricate arbitrary objects, as evidenced by the abundant literature on the subject [Schwartzburg and Pauly 2011; McCrae et al. 2011; Hildebrand et al. 2012; Schwartzburg and Pauly 2013; Cignoni et al. 2014; McCrae et al. 2014].

2.1.3 Enforcing Printing Constraints

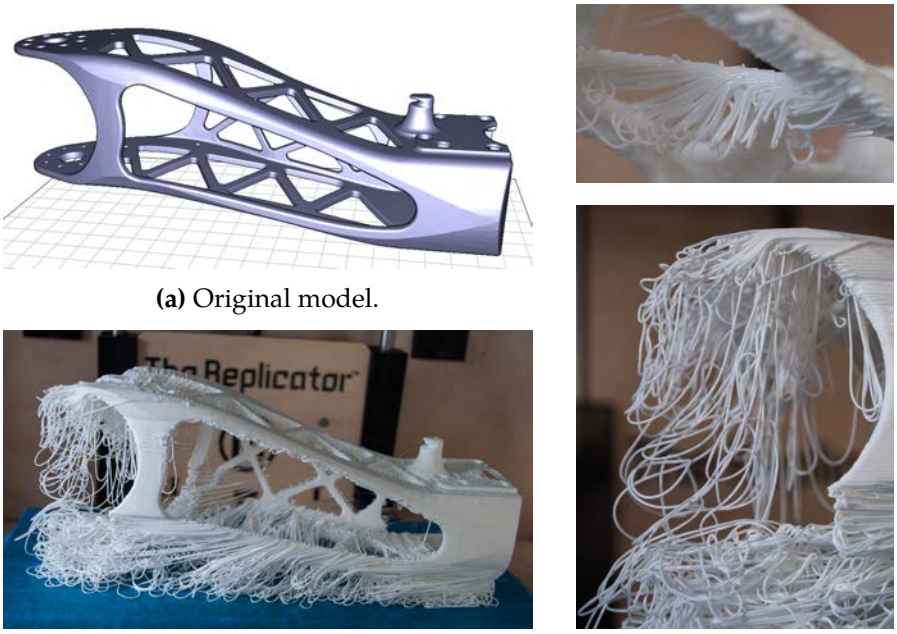
Overhang and Supports

On filament printers, perhaps one of the most restricting constraint in terms of design is in fact due to the limited slope that can be achieved when stacking multiple layers together. To overcome this limitation, the standard practice is to print the object with an external support structure, removed after the print. Note that these support structures can be very tedious to clean up, and they often alter the appearance of the object surface. Thus, notwithstanding material costs, it is advisable to reduce as much as possible the amount of supports that is needed to print an object.

If an object is printed without support material on FDM printers, chances are it will lead to an intriguing result, often quite far from the intended one. Figure 2.5

illustrate such a failure case on the upper leg of the Poppy robot[1]. Note however, that there are a number of reasons as to why a print can fail, besides improper support structures. There is even a Flickr gallery entitled "The Art of 3D Print Failure"[2] dedicated to this subject.

In order to generate a support structure for a given shape, one usually proceeds in two steps: first, detect the actual parts of the object that need to be supported. Second, generate the actual support geometry. Finally, one can also decide to orient of deform the original model to reduce the need for supports in the first place. This approach is known as *support slimming*.



(a) Original model.

(b) Printed result.

(c) Zoom.

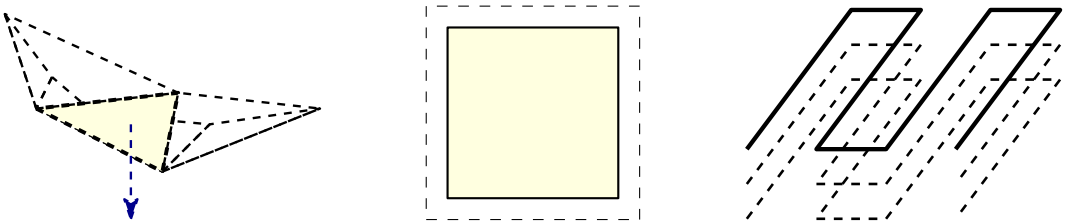
Figure 2.5 – Example of a print failure on the leg of the Poppy robot. Without support structures, the object cannot be fabricated in this orientation on FDM printers due to the overhang constraint.

Determining Support Requirements. Support generation algorithms start by determining the surfaces in need for support. A first family of approaches consider the down-facing facets of the input mesh having an angle too steep to print correctly [Allen and Dutta 1995; Alexander et al. 1998]. A second family of approaches consist in performing a boolean difference between two successive slices [Allison et al. 1988; Chalasani et al. 1995; Huang et al. 2009a]. This generally leads to a compact set of points to be supported. Eggers and Renap [2007] select a subset by down-sampling. In Section 3.1, we follow a similar approach to [Chalasani et al. 1995], considering

[1] www.poppy-project.org

[2] https://www.flickr.com/groups/3d-print-failures/pool/

whether the plastic deposition paths are correctly supported by the layers beneath. Those three different principles are illustrated Figure 2.6. Finally, Telea and Jalba [2011] study the printability of models using a voxel representation of the volume. Note that a caveat of support detection via boolean operations is that it can falsely detect unsupported regions as supported (see Figure 1.8 in [Huang et al. 2014a]).



(a) Triangle-based. **(b)** Morphology-based. **(c)** Printing paths.

Figure 2.6 – Three approaches to the support detection problem:
(a) Considering the angles of every facet in a triangle mesh.
(b) Computing boolean operations between successive layers.
(c) Evaluating the actual printing paths on filament printers.

Generating the Support Geometry. Once the parts requiring support are determined, the support geometry is computed. The standard approach consists in extruding downward the mesh facets requiring support, thus defining a *support volume*. The support volume is usually printed with a weak infill pattern (*KISSlicer*, *MakerWare*, [Strano et al. 2013]). The support is manually removed by breaking it apart from the object. Soluble material can also be used on multiple material printers [Kritchman et al. 2008]. Printing the support volume uses a significant amount of material and print time, but is very reliable: the support typically has a large area of contact with both the part and the print bed, ensuring the print stability in most cases. The volume is large enough to print without difficulty.

A number of approaches modify the support volume to reduce its size. Huang et al. [2009b] use sloped walls instead of straight walls for the sides, shrinking the support volumes in their middle sections. Heide [2011] also reduces the support volume by decreasing its size and complexity as the distance below the supported model increases. More recently, Jin et al. [2015] proposed a support generation approach based of boolean operations between successive layers, in an manner which is closely related to the method presented is [Hornus et al. 2015].

MeshMixer by *Autodesk* deviates from the support volume and instead builds a thin structure supporting the part in a sparse, limited number of points. *MeshMixer* automatically generates an initial support resembling a tree. While the precise algorithm, to the best of our knowledge, has not been published as an article, some insights can be found in their SIGGRAPH 2014 talk [Schmidt and Umetani 2014]. This work elegantly shows that a very sparse structure can effectively serve as a support. *MeshMixer* however often requires the user to fix up the initial structure (see

comments on Thingiverse, `thing:131054`), and the slanted trees sometimes suffer from print reliability issues (see Figure 3.18b). In contrast, our bridges presented Section 3.1 offer similar grouping properties as a tree but print more reliably. A concurrent work, presented by Vanek et al. [2014a], features support structures very similar to *MeshMixer*.

Wang et al. [2013] optimize truss structures for the primary purpose of strengthening 3D printed objects, and extend their approach for support generation. Support beams are added by tracing rays downwards, within a tolerance cone ensuring that the result is printable. The beams are not grouped, missing an opportunity to reduce print time and material usage.

Eggers and Renap [2007] propose to form a support structure by starting from a regular rhombus mesh filling the print bed. The 3D model is subtracted from the initial structure, removing intersected mesh edges. Points requiring support are attached to the mesh by downward angled beams. A number of heuristics are proposed to reduce the number of beams in the support structure. This approach, however, has to ensure that large enough columns are formed so that the support mesh remains printable. In contrast, in Section 3.1, we optimize for thin elongated bridges that are guaranteed to remain printable. Our approach does not suffer from the orientation bias resulting from canceling edges in a pre-existing mesh.

Photoshop CC includes support generation for 3D printing. Available screenshots reveal that square-section pillars are grown from the ground with a hierarchical tree structure to connect to the surface.

Orientation and Support Slimming. Several approaches have been proposed to find a model orientation reducing the amount of supports required before printing [Frank and Fadel 1995; Allen and Dutta 1995; Cheng et al. 1995; Alexander et al. 1998; Majhi et al. 1999; Ezair et al. 2015; Morgan et al. 2016]. Note that other criteria for choosing model orientation can also be taken into account, in particular mechanical stress [Umetani and Schmidt 2013; Ulu et al. 2015], visual appearance [Zhang et al. 2015d], or upright correction [Wang et al. 2014a]. Another possibility is to deform the model to further reduce the need for support material [Hu et al. 2015].

We do not consider this issue in our work Section 3.1, and assume that the part orientation has been fixed by the user considering other criteria (stepping error, mechanical robustness, aesthetics of filament orientation).

Thickness Control and Morphological Operations

Another important printing constraint, which is common to most printing technologies, is length-scale control. On filament printers, the minimum thickness of solid parts is mostly limited by the nozzle width — although it is possible to print smaller features by controlling the flow of plastic. On SLA and SLS printers, as well as laser

cutters, the width of the laser beam, or resolution of the projector, is limiting the size of both solid and empty regions.

To ensure manufacturability of a 3D model, it is thus imperative to control its minimum feature size and, in some applications, its minimum hole size. To this end, *morphological operations*, such as dilation, erosion, opening and closing are extremely useful [Williams and Rossignac 2005]. Figure 2.7 illustrates the effect of different morphological operations on a given shape. In the context of digital fabrication and computer graphics, methods have been proposed to compute offsets of 3D surfaces [Chen and Wang 2011; Liu and Wang 2011; Wang and Manocha 2013; Martínez et al. 2015b]. A closely related problem to consider is the thickening of a flat 2D surface. The offsetting occurs either *inwards*, or *outwards* the input surface, and one should take care to avoid self-intersections [Wang and Chen 2013]. In Section 3.2, I will introduce an novel algorithm for fast morphological operations in 2D images.

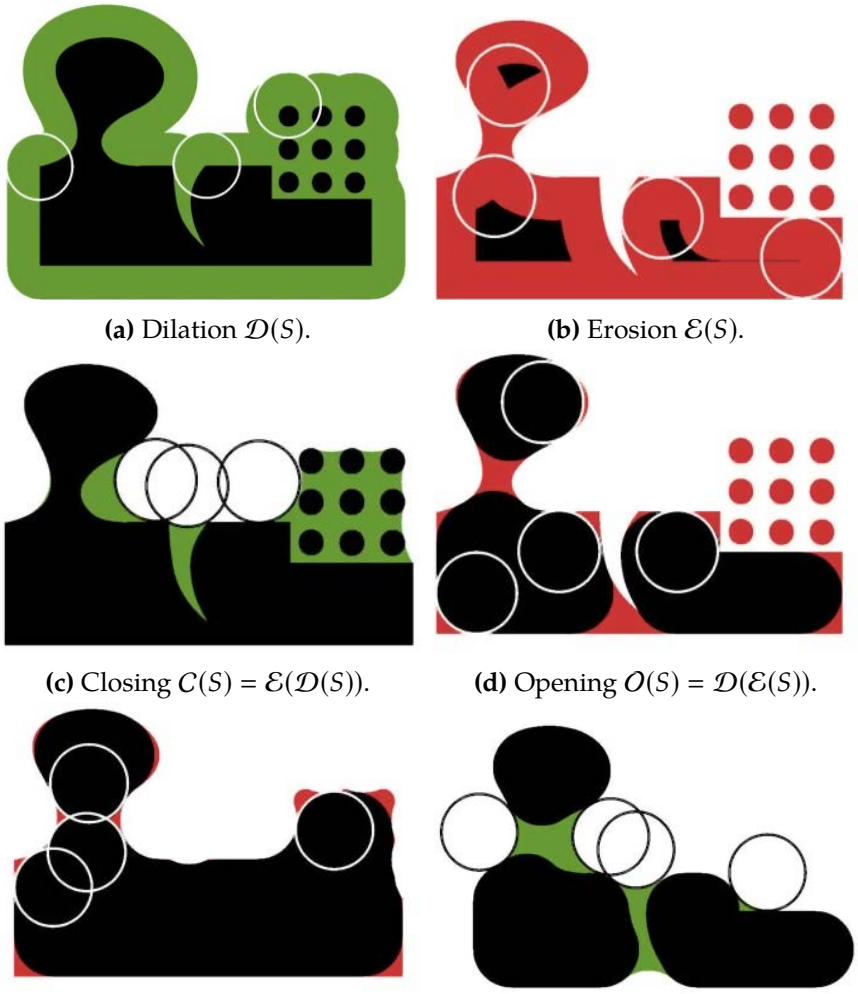
Other Constraints

Part Stability. Parts may topple during printing, either due to weight imbalance [Chalasani et al. 1995] or under the friction forces of the printing head. The literature on this subject is very scarce, and I am not aware of any approach that considers the stability of partially printed parts, *at all stages of the printing process*, and propose to correct it in an automatic way. Indeed, while an object may be balanced once printed, its — possibly disconnected — subparts may not be stable before all layers are stacked together.

Our support generation algorithm, presented in Section 3.1, takes into account the partial stability during the printing process, and we propose to adapt our support structures to take this phenomenon into account. Note that the overall static stability of a shape, once the object is printed entirely, is a key property to consider when designing an object for fabrication. This aspect of the design process will be further discussed in Section 2.2.2.

Print Volume and Segmentation. In Section 2.1.2, we have presented segmentation techniques to improve the surface quality the printed shapes. However, the most common reason where one needs to segment a shape, is perhaps to fabricate an object larger than the print volume allowed by the printer. Not only is it necessary to decompose the input shape into multiple parts, but those parts need to be arranged and packed into the allocated print volume as efficiently as possible, to reduce the number of times one has to operate the printer.

Several approaches have been proposed to decompose and pack a model into a small number of parts. [Hao et al. 2011] decompose a 3D model according to its surface curvature. [Luo et al. 2012] proposed an approach to partition a shape into multiple parts so that they fit into a target build volume. Vanek et al. [2014b] and Chen et al. [2015b] consider the joint problem of decomposing a shape and packing the parts



(a) Dilation $\mathcal{D}(S)$.

(b) Erosion $\mathcal{E}(S)$.

(c) Closing $C(S) = \mathcal{E}(\mathcal{D}(S))$.

(d) Opening $O(S) = \mathcal{D}(\mathcal{E}(S))$.

(e) Inner-regularization $O(C(S))$.

(f) Outer-regularization $C(O(S))$.

Figure 2.7 – Effects of different morphological operations on a simple shape. The input shape is shown in black in (a). Image: [Williams and Rossignac 2005].

into the given build volume. Attene [2015] explores a user-controllable trade-off between the number of parts created by the segmentation, and the tightness of the resulting packing. Yao et al. [2015] consider structural stress among other measures to assess the quality of their segmentation. Song et al. [2016] propose an approach to fabricate objects much larger than standard 3D printed build volume by combining 3D printed surface parts with a core skeleton obtained through other methods (e.g. laser-cutting). Note that other methods offer to decompose a shape into interlocking elements, which increases the stability of the assembled object [Xin et al. 2011; Song et al. 2012, 2015].

2.2 Shape Design for Fabrication

As Section 2.1 was dedicated to the presentation of *geometrical* constraints, such as minimum thickness, and overhangs, it is now natural to discuss *structural* constraints that should be considered when designing 3D models for fabrication. A structural constraint denotes any shape characteristic that does not impede the manufacturing process, yet renders the fabricated object too fragile to be manipulated. A model that violates geometrical constraints will not print correctly. However, a model that violates structural constraints but not geometrical ones will print correctly, but is more likely to break under normal manipulation conditions.

In this section, we are particularly interested in how design tools can assist the user in analyzing, editing, or synthesizing 3D shapes in a manner that is suitable for fabrication. Such tools should consider both the geometrical and structural constraints mentioned above, and either provide automatic corrections for a particular problem, or provide an interactive interface to assist the user in fixing the problem manually.

The rest of this section is organized as follows. In Section 2.2.1, we first present methods for the analysis and simulation of the elastic behavior of 3D printed shapes. Section 2.2.2 focuses on tools developed to edit and optimize existing shapes for a particular purpose. Finally, in Section 2.2.3, we extend the discussion to methods that synthesize new content in a fabrication-aware fashion, either from an incomplete model definition, or from a more abstract description. This approach proposing computer-assisted design tools is sometimes referred to as *computational design*.

2.2.1 Structural Shape Analysis

Given a 3D model to be printed, several works have tried to answer the crucial question of whether the shape is mechanically sound, or if it will be too fragile once printed. The initial work of [Telea and Jalba 2011], mentioned in Section 2.1.3, is a first automated attempt in this direction, but they only analyze *geometric* constraints, namely the minimum thickness.

Stava et al. [2012] describe what is probably the first automated solution dedicated to the analysis of *structural* fragility in 3D printed models. Their algorithm computes stress and displacements on a tetrahedral mesh of the model using the Finite Element Method. They consider two types of loads. The first load is gravity force, which corresponds to the weight of the object. The second load is grip forces, caused by the pressure of fingers when a user manipulates the object. After analysis, the authors propose several ways to reinforce a model in order to meet the structural requirements. The first option is local thickening, based on an analysis of the medial axis in regions which are too fragile. The second option is adding struts between two points of the model. The struts are added using an heuristic compromise between their mechanical efficiency, and their impact on the object appearance. The last option, to alleviate the effect of gravity, is hollowing the interior of the shape.

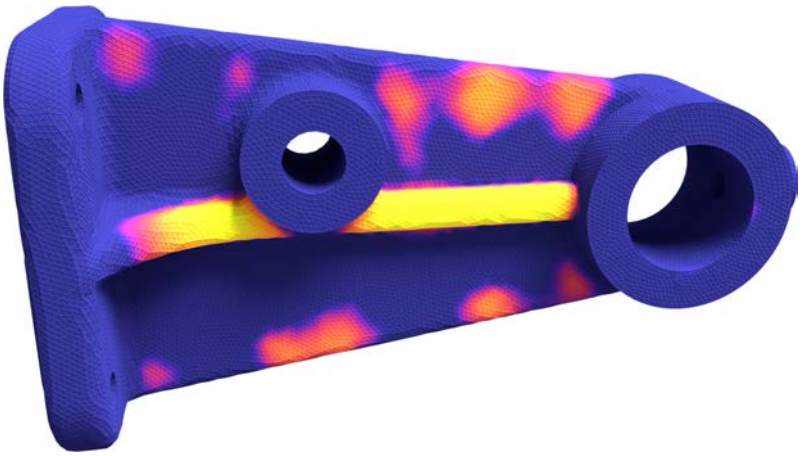
A subsequent work by Zhou et al. [2013] proposed a more robust, and more generic approach to shape analysis compared to [Stava et al. 2012]. In [Zhou et al. 2013], the authors study vibration modes of the input 3D shape, to determine which region of the model is the weakest. It does so by computing the maximum possible stress under any pressure distribution on the surface. By making some assumptions on the pressure distribution, such as bounding the maximum total pressure, or limiting the pointwise maximum pressure, the system admits a solution. The computation is also sped up by relying on a coarse tetrahedral mesh at earlier stages of the algorithm. While the paper showcases impressive results, it remains difficult to suggest design changes that will alleviate all identified problems. An illustration is given Figure 2.8, where a bearing bracket is analyzed using [Zhou et al. 2013]. The resulting weakness map can then be used to compute a spatially-varying microstructure that is more lightweight in the less fragile regions, thus saving material. More recently, Langlois et al. [2016] proposed a stochastic approach to analyze the structural weaknesses of an object, by predicting the probability of failure in the different parts of the model.

In order to achieve the real-time analysis performances necessary for interactive modeling purposes, further work is required. In [Umetani and Schmidt 2013], the authors proposed a different analysis technique suitable for interactive applications. It their work, they consider bending forces to analyze cross sections of a 3D model, which are modeled according to the Euler-Bernoulli beam theory. By computing bending moments, they are able to determine quickly which cross section of an object is more likely to break, without the need to solve an expensive FEM equation. This is especially important in the context of filament 3D printers, where printed objects have much stronger cohesion in the XY direction, but can delaminate in the Z direction. It is thus desirable to orient weak cross sections of the model orthogonaly to the Z direction of the print. The method presented in [Umetani and Schmidt 2013] has some limitations, in particular behaviors which are not well captured by the underlying theoretical framework, such as buckling, certain problematic boundary conditions — e.g. a bar held by both extremities —, or round intricate objects — e.g. Voronoi Sphere (`thing:221740`).

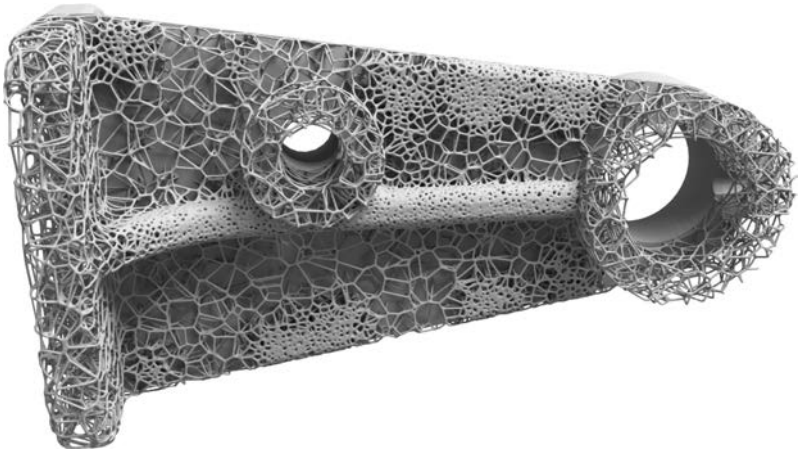
In another effort to provide real-time simulation feedback for interactive shape modeling, Xie et al. [2015] proposed a method based on domain decomposition and local computation, to provide interactive information about the edited model. Finally, in a recent paper, Xu et al. [2016] augment the cross sectional analysis strategy of [Umetani and Schmidt 2013] with information about the curve skeleton of the analyzed shape, seemingly producing more accurate results. They also rely on the shape skeleton to propose corrections to the model via e.g. local thickening.

2.2.2 Design Editing for Fabrication

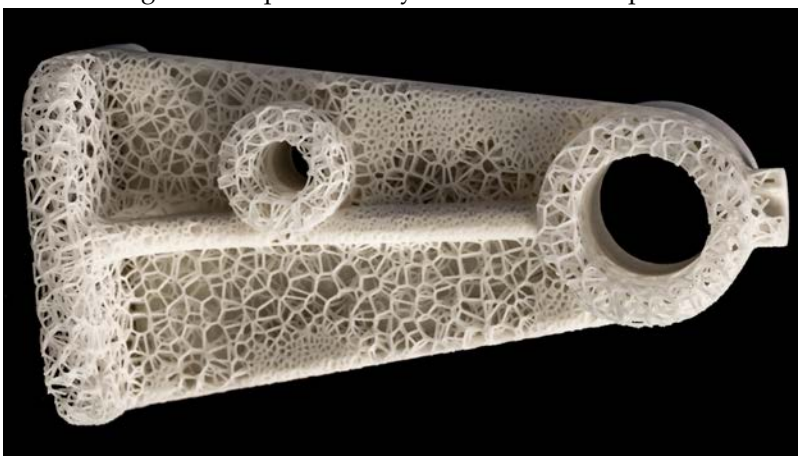
Ascertaining the printability of a given shape is an essential step in the printing pipeline. However, this does not always suggest what is the best way to edit the design of an existing model in order to achieve a desired functional goal. For



(a) Potential defects can be detected by analyzing vibration modes [Zhou et al. 2013].



(b) Microstructure generated procedurally with our method presented Section 5.1.



(c) Model printed on a *ZCorp 450*.

Figure 2.8 – To reduce the material cost of a print, a lightweight internal structure can be devised, with a higher density in the more fragile regions. Model: `thing:994788`.

example, one may want to print non-assembly articulated models [Bächer et al. 2012; Calì et al. 2012], balance a shape so that it can stand in an arbitrary orientation [Prévost et al. 2013], or optimize the interior of an object to maximize its strength-to-weight ratio [Lu et al. 2014]. In this subsection, we present some of the related work most relevant to this thesis, in particular for the purpose of shape balancing, and multi-purpose criteria that can arise in shape editing.

Balancing Shapes

The problem of balancing shape has gained a distinct interest in the computer graphics community since the publication of *Make It Stand* [Prévost et al. 2013]. The idea of the paper is that, given a input shape and an arbitrary orientation, one must compute a modified shape that, once printed, can stand in static equilibrium in the given orientation. The modifications to the input shape must minimal, so as to preserve its appearance. While conditions for static equilibrium are well known — the center of mass of the shape needs to vertically project in the 2D convex hull of its points in contact with the ground —, the idea of an automatic process targeted for digital fabrication really spurred a new trend. An illustration of the process is shown Figure 2.9.

On the technical side, Prévost et al. [2013] and most of its follow-ups rely on similar principles. To remove excessive interior weights, hollowing is carried out on a voxelized version of the volume mesh [Prévost et al. 2013; Bächer et al. 2014], or in directly in a tetrahedral mesh [Christiansen et al. 2015b]. Rotations or small local deformations can also be applied to better achieve the target objective. In a recent paper, Wu et al. [2016b] present a different framework to optimize the material distribution of interior volumes. Their approach is based on ray-reps representations, such as the dexels described in Section 2.1.2. Ray-reps are an efficient representation of volumetric data that are very simple to process and update, hence their interest for interior volume design.

While Prévost et al. [2013] were concerned only with the static equilibrium of standing or suspended shapes, subsequent work has explored other balancing objectives. Bächer et al. [2014] optimize the rotational stability of spinning tops, by aiming for the lowest possible center of mass. Christiansen et al. [2015b] achieve the same objective as [Prévost et al. 2013], but with a different approach: instead of deforming the surface shape, they propose to rotate the model around its base, producing less noticeable changes, at the expense of more limited possibilities. Other works seek to optimize the floating properties of printed objects [Wang and Whiting 2016], or create roly-poly toy [Zhao et al. 2016a].

Please note that static equilibrium of shapes has also been extensively studied in the context of architectural design and masonry buildings. Some references in the computer graphics literature include [Whiting et al. 2009; Vouga et al. 2012; Whiting et al. 2012; Song et al. 2013; Panozzo et al. 2013; Deuss et al. 2014; Pietroni et al. 2015;

Frick et al. 2015]. A recent study comparing finite element analysis and stability analysis for masonry buildings is presented in [Shin et al. 2016].

(a) Left: Input model. **Right:** Balanced model. **(b)** Printed result standing.

Figure 2.9 – Automatic shape balancing from [Prévost et al. 2013]. An unoptimized model is given as input to the system, which is allowed to slightly deform its surface, and carve its interior, to produce a balanced shape under the prescribed scenario (upright direction, contact points with the ground).

Multi-Objective Optimization

In an effort to develop an efficient shape optimization framework for typical problems that arise in the context of digital fabrication, Musialski et al. [2015, 2016] presented two generic optimization procedure, and demonstrate their effectiveness on a variety of specific problems.

In their first paper, Musialski et al. [2015] cast the shape optimization problem as a sizing problem, where the thickness of an offset surface is to be determined according to a certain objective function. By parameterizing the offset displacements using harmonic manifolds — the equivalent of Fourier transforms for meshes [Vallet and Lévy 2008] —, and solving the optimization problem by projecting in the subspace spanned by only the first few vectors of this new harmonic manifold basis, they are able to quickly solve problems with a large number of variables.

Their method was later extended in [Musialski et al. 2016]. This second approach is designed for multi-objective optimization, where the goal is to minimize a weighted sum of different shape properties φ_i. The shape is parameterized by a set of parameters $\boldsymbol{\alpha}$, which constitute the variables of the system. The key insight of [Musialski et al. 2016] is to perform a dimensionality reduction that maps $\boldsymbol{\alpha}$ to a set of reduced parameters $\boldsymbol{\beta}$, which decorrelates their influence on the shape properties φ_i. A linear mapping $\boldsymbol{B}$ such that $\boldsymbol{\alpha} = \boldsymbol{B}\boldsymbol{\beta}$, is computed. The matrix $\boldsymbol{B}$ is computed from the current state of the system, using the gradients $\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{\alpha}}$.

The algorithm then runs two nested loops. In each outer iteration, the matrix $\boldsymbol{B}$ is recomputed, so as to decorrelate the variables $\boldsymbol{\alpha}$, and reduce the dimensionality of the problem. In each inner iteration, the reduced problem is solved using classical gradient-based methods like Quasi-Newton or Levenberg-Marquardt. The

algorithm is applicable to non-linear and constrained optimization problems, and does not rely on a harmonic manifold basis to perform the dimensionality reduction. We note however that when there is only one shape property φ_i to optimize, the outer loop simply amounts to a dimension reduction much like in the previous paper [Musialski et al. 2015].

2.2.3 Computational Design, Shape Synthesis and Completion

The term *computational design* encompasses a broad category of techniques, that aim at integrating physical considerations into all sorts of design tools. While an exhaustive overview is beyond the scope of this thesis, we present here a selection that is the most relevant to our work. In particular, we focus on tools for shape synthesis and shape completion, where new content is generated from more high-level, possibly incomplete, specifications. The work presented in this thesis, in particular Chapter 4, can be seen as an instance of computational design, where physical constraints are taken into account in the design process, and optimized together while considering the appearance of the synthesized shapes. Synthesis methods for appearance optimization, but without considerations for fabrication, will be discussed next in Section 2.3.

Dedicated Tools. A recent line of work in computer graphics that concerns — for lack of a better term —, dedicated tools for *computational design* purposes. Tackling one goal at a time, several specialized solutions have been proposed. For example, Umetani et al. [2011] developed an interactive tool for garment editing that directly integrates physical simulation. In [Umetani et al. 2012], the authors explore a range of physically valid furniture designs. More generally, Shugrina et al. [2015] developed a tool for the intuitive exploration over a set of physically sound parametric models. In [Umetani et al. 2014], a data-driven method for the drawing of model airplanes is presented. Martin et al. [2015] present an application for creating functional kite designs. Applications to sound are also emerging, e.g. for the design of musical instruments with custom shapes [Umetani et al. 2010; Bharaj et al. 2015; Umetani et al. 2016].

Mechanical Characters. The design of animated mechanisms reproducing a given movement has been investigated in [Coros et al. 2013; Ceylan et al. 2013; Thomaszewski et al. 2014]. Bächer et al. [2012] and Calì et al. [2012] study how to model and fabricate articulated characters without assembly — the articulated model is printed as a single piece. The challenge lies in designing and placing the joints on a given, non-articulated input shape. Megaro et al. [2014] present an intuitive interface to model 2D mechanisms. Hergel and Lefebvre [2015] optimize the 3D layout of a mechanism, given partial information about its 2D configuration, while improving the robustness of the fabricated design.

Elastic Shapes. Several methods have been proposed to synthesize objects that can be deformed or animated with flexible materials. Skouras et al. [2012] design rubber balloons which take a prescribed shape once inflated. Skouras et al. [2013] animate characters with a small number of actuators, so that it can assume a target shape — the algorithm computes the number and position of the actuators, as well as material distribution that will allow the desired motion (Figure 2.10). Chen et al. [2014] solve an inverse problem to compute the rest shape of a deformable model that will assume the desired shape under the effect of gravity once printed. Note that inverse elastic shape design has been previously employed in the context of animation and hair simulation [Derouet-Jourdan et al. 2010, 2013]. More recently, the design of flexible rod meshes has been investigated in [Pérez et al. 2015].

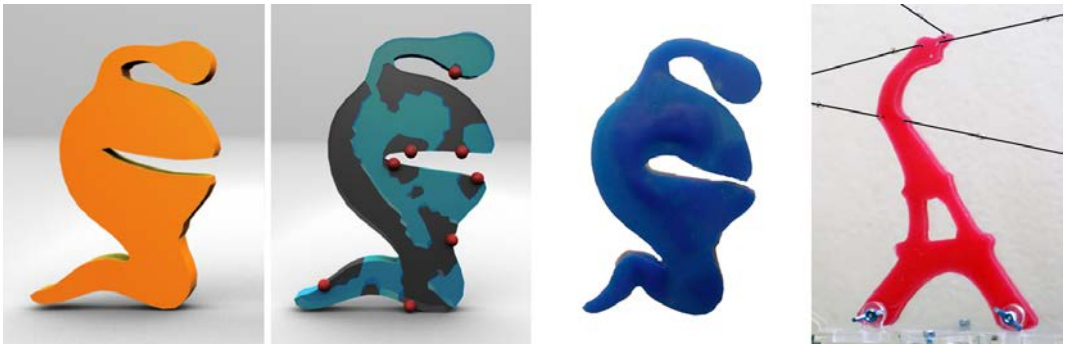


Figure 2.10 – Synthesis of deformable characters [Skouras et al. 2013]. A material distribution mixing two different materials is optimized, so the fabricated shape can deform according to a prescribed scenario. The characters are then animated using a small number of actuators.

2.3 By-Example Shape and Texture Synthesis

An important part of this thesis is dedicated to controlling the appearance of an object during the model synthesis, and how it can affect the mechanical properties of the printed structures. Appearance control can be achieved by multitude of methods, two of which are by-example approaches, and procedural evaluation of implicit functions.

By-Example Synthesis. The goal of by-example synthesis methods is to produce new content from a user-given input exemplar, in a way that preserves its appearance. One of the first application of by-example synthesis is texture synthesis [Efros and Leung 1999; Wei and Levoy 2000], where an low-storage input image is used to generate a high-resolution texture. If the algorithm can compute its result in real-time, it can be integrated into video games and other interactive applications.

There are different ways to classify texture synthesis techniques. One can distinguish whether the synthesis is performed in 2D, on a surface (embedded in 3D space), or directly in 3D (volume synthesis). One can also distinguish between raster texture synthesis (where the exemplar is a 2D or 3D array of pixels), and vector texture synthesis (where we manipulate polygonal lines, Bezier curves, or 3D meshes instead of regular grids of pixels).

Despite those different categories, most by-example texture synthesis algorithms can be decomposed into three essential steps. First, the output domain is filled with an initial guess — random values, or patches copied from the input image. Second, neighborhood matching is performed: for every pixel in the output image, find a good location in the input that shares the most similar neighborhood. Third, update the output pixels according to some criteria — vote from the adjacent pixels, gradient direction of an energy function, etc. In any case, the appearance of the output image is evaluated locally, by calculating how much each pair of matched input/output neighborhoods are similar. This is an important property, as it means the usual texture synthesis methods cannot reproduce images with a highly structured content (e.g. a building), for which further work is necessary. Figure 2.11 illustrates the principle of 2D texture synthesis, and Figure 2.12 presents different strategies for the neighborhood search.

By-example texture synthesis techniques were the first approaches to try to reproduce appearance based on an input exemplar. Later, other methods have emerged, that attempt to capture and reproduce more structured data, such as geometric patterns [Bhat et al. 2004; Lagae et al. 2005; Zhou et al. 2006], or even entire 3D models [Merrell 2007]. We build upon these works, and present in Chapter 4 several example-based shape synthesis methods that not only generate structured objects, but also take into account their mechanical properties.

Procedural Synthesis. Another approach for content generation with controllable appearance is achieved via *procedural* synthesis methods. While the term itself is very generic, we restrict our definition of procedural approaches to functions of space $\mathcal{F}(\mathbf{x})$ that can be evaluated efficiently and with a local support. This generally includes the evaluation of mathematical functions defined implicitly, fractals, shape grammars, etc. In computer graphics, the first methods dedicated to the synthesis of solid (3D) textures were using procedural functions [Peachey 1985; Perlin 1985].

Shape grammars, which we do not discuss further in this thesis, are another powerful tool for the procedural synthesis of complex structures. Since the original development of L-systems for the modeling of plants [Lindenmayer 1968a,b; Prusinkiewicz et al. 1988], shape grammars have proven an effective approach to generate highly structured content, such as urban landscapes [Vanegas et al. 2010], or buildings that can be 3D printed [Kalojanov et al. 2016].

Note that procedural and by-example approaches are not mutually exclusive: a procedural synthesis method can rely on exemplar images and aim at synthesizing

visually similar content, e.g. to produce an "infinite zoom" effect with on-the-fly texture synthesis [Han et al. 2008; Vanhoey et al. 2013]. In this thesis, we propose in Section 5.1 a procedural approach for the generation of microstructures with controllable elastic properties inside a given volume.

Summary. The rest of this section is organized as follows. In Section 2.3.1, we present existing techniques for 2D content synthesis, whether it is restricted to the 2D plane, or to a surface embedded in 3D. The focus is put on by-example synthesis methods, which encompasses geometric texture synthesis and style transfer. In Section 2.3.2, we extent the discussion to volume synthesis methods, again with a stronger focus on by-example approaches. Note that we only discuss techniques that are the most relevant to our work, or that we find inspiring in the context of fabrication and structural optimization. For a more comprehensive survey up to 2009, regarding by-example texture synthesis methods in general, the interested reader is referred to [Wei et al. 2009]. Finally, in Section 2.3.3, we move the discussion from *unstructured* to *structured* content generation, while keeping the emphasis of by-example synthesis methods. Structured content generation imposes additional constraints, which are not present in unstructured pixel-based synthesis methods.

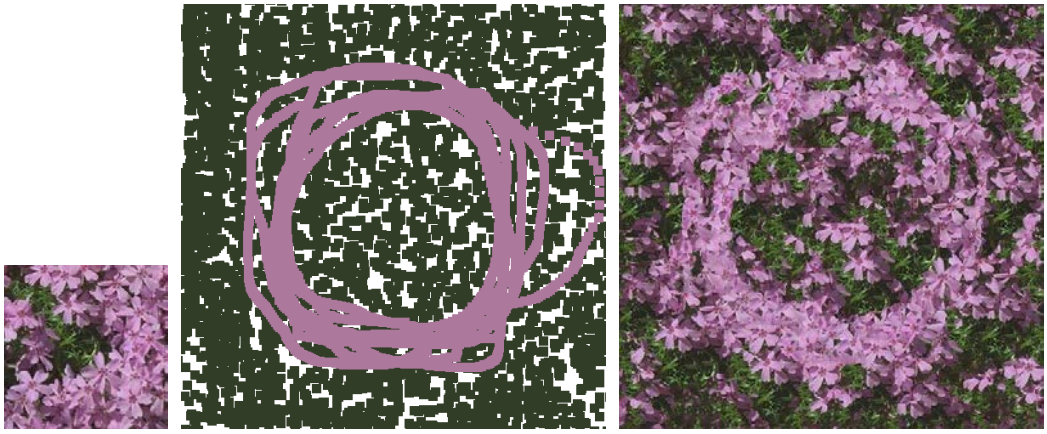


Figure 2.11 – 2D texture synthesis [Ashikhmin 2001]. An input exemplar (left) is used to synthesize a larger image (right). The synthesis can be driven by a feature map painted by the user (middle).

2.3.1 Surface Synthesis

By-Example 2D Texture Synthesis

The initial methods for by-example texture synthesis are based on Markov Random Fields [Efros and Leung 1999; Wei and Levoy 2000]: a probabilistic model is defined by sampling neighborhoods from the exemplar. These approaches are not trivially amenable to our context due to the stochastic nature of the optimization process.

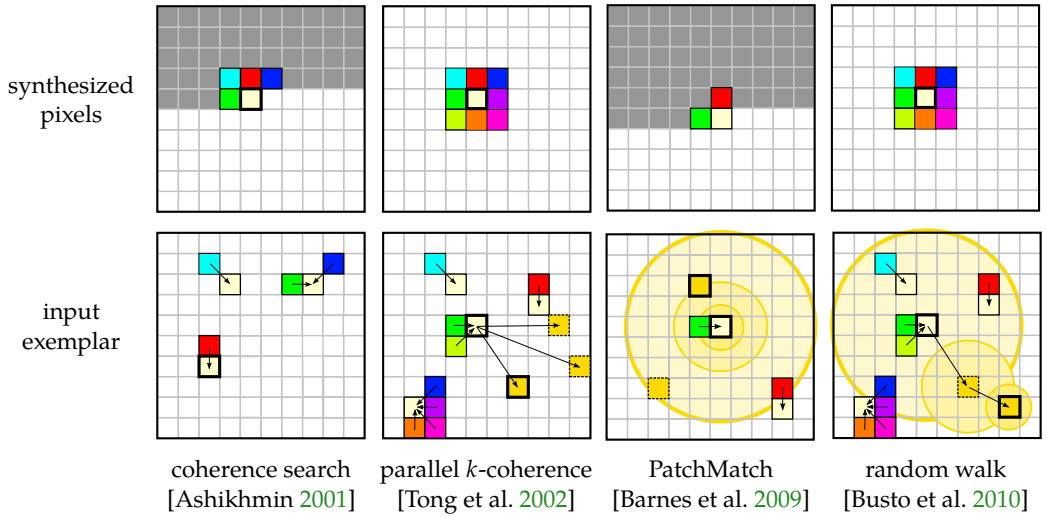


Figure 2.12 – Neighborhood search. To find the input neighborhood the most to a given output neighborhood, different acceleration strategies can be employed. In addition to speeding up the matching step, restricting the neighborhood searches to coherent candidates can also prevent the synthesizer from getting stuck in a trivial solution. (Image courtesy of [Busto et al. 2010].)

Kwatra et al. [2005] proposed a different point of view of the problem which is based on formulating an energy — the pixels of the output image being the variables. By optimizing this energy a new image is synthesized which resembles the example. We adopt this point of view in Section 4.2 to define the appearance energy relating the produced shape to the exemplar.

In recent work, Wu et al. [2014] present a texture synthesis method based on a level-set representation of the texture. In [Kaspar et al. 2015], the authors present a self-tuning, non parametric method to generate various high resolution textures.

Few works have considered both texture synthesis and fabrication. [Zhou et al. 2014b] synthesize patterns along curves while precisely controlling their topology. The results can then be laser-cut or 3D printed as they form singly-connected objects. It is however essentially a one-dimensional synthesizer and the approach does not consider the structural properties of the generated objects.

Finally, closer to our goals, [Zhou et al. 2014b] synthesize fabricable patterns along a curve, from an example. This approach is specifically designed to synthesize patterns with controlled topology.

Synthesizability of Textures Exemplars. Not all images provide good exemplars suited for texture synthesis techniques. Dai et al. [2014] discuss the problem of finding good candidate images for synthesis purposes. Categorization of textures is also discussed in [Lefebvre 2014], and in Section 4.1.5. Stochastic textures which

exhibit little structural elements will be the easiest to synthesize, while images with very distinguishable features (such as letters of an alphabet) will be very hard to synthesize without specialized algorithms.

By-Example Texture Synthesis on Surfaces

The principle of texture synthesis on surface is illustrated Figure 2.13. Turk [2001] and Wei and Levoy [2001] adapt the image-based approaches comparing small neighborhoods of colors to work along a mesh surface, considering densely tessellated meshes with per-vertex colors. Tong et al. [2002] proceed similarly but synthesize in every vertex a texton label capturing a BTF[1] appearance. A drawback of these techniques are the dependency on tessellation and the resampling required when working with distorted neighborhoods along the surface. Ying et al. [2001] synthesize the texture in a parametric space: the surface is divided into planar charts into which synthesis is performed, and the result is mapped back to the surface. A similar methodology is used by Lefebvre and Hoppe [2006] in a scheme that performs parallel texture synthesis into the pixels of a texture atlas. These approaches are relatively complex as they have to cope with parametric distortions, discontinuities at chart boundaries, and sampling issues.

Praun et al. [2000] perform synthesis along a surface by applying many texture patches with irregular boundaries so has to give the illusion of a continuous texture. Soler et al. [2002] optimize a set of texture coordinates for the triangles with the objective of producing a visually seamless texture along the surface. Recently, Gagnon et al. [2016] proposed an atlas-based texture synthesis method for texturing fluids.

To create texture maps of scanned 3D models, Lempitsky and Ivanov [2007] and Gal et al. [2010] align photographs with the geometry an then formulate a labeling problem to select one image for each triangle. In a more recent work, Pagés et al. [2014] create a texture atlas from multiple photographs of the model. Our synthesizer presented Section 4.1 takes a labeling view similar to [Lempitsky and Ivanov 2007; Gal et al. 2010], using randomly positioned texture planes and voxels.

All these approaches only generate colors along the surface and do not modify the underlying geometry of the model.

By-Example Geometry Synthesis

A number of approaches have been proposed to go beyond color synthesis, and generate geometric details along the surface of an object.

Bhat et al. [2004] synthesize geometric details using the by-analogy approach initially developed for images [Hertzmann et al. 2001]. The geometry is captured by voxels

[1]Bidirectional Texture Function, a 6D function modeling spatially varying reflectance properties.

Figure 2.13 – Texture synthesis on surface [Lefebvre and Hoppe 2006]. A texture atlas (middle) is synthesized automatically from an input exemplar (left), and then mapped onto the target surface (right). Note the distortions introduced in the texture atlas, even though the resulting surface appears undistorted. Dealing with distortions through texture mapping and multiple indirections is a tedious step unavoidable in most texture synthesis methods.

and a distance field, and therefore the synthesizer is free to carve and sculpt the object. The scheme is based on voxel neighborhoods and therefore requires the input exemplar to be a small 3D pattern with geometric details. Lagae et al. [2005] perform geometry synthesis of 3D patterns by comparing blocks of voxels in a distance field.

All the aforementioned approaches employ a 3D version of the 2D neighborhood matching of texture synthesis. Comparing voxel neighborhoods in a volume throughout the object is computationally expensive, as it requires to process a significant amount of volume data through the multi-scale neighborhood dependencies. It contrast, our scheme presented Section 4.1 performs synthesis in a set of voxels representing the surface, but it maintains a one-voxel thickness across all scales, and does not involve comparing 3D neighborhoods of voxels.

Zhou et al. [2006] synthesize a detailed mesh around a guiding surface mesh by stitching together geometric elements cut out from an input example geometry. The elements are deformed, aligned and stitched to produce a continuous result which is grown in a parametric domain around the model. Impressive patterns are obtained, the main drawbacks being the need to define elements in an input mesh and the geometric distortion in high curvature regions which would make fabrication delicate. Another approach, presented in [Li et al. 2011], relies on shape grammars instead of exemplars, to provide control over the synthesized geometry. These approaches are not designed to guarantee connectivity or structural soundness.

Style Transfer

Style transfer is a variant of by-example synthesis where the process is guided to enrich existing content with details. Hertzmann et al. [2001] pioneered this idea by proposing to transfer details specified by a pair of images A:A' to produce an image B' from a different source image B. Recent approaches have explored how to exchange styles within and across collections of shapes [Xu et al. 2010, 2012; Li et al. 2013; Han et al. 2014; Ma et al. 2014]. These approaches typically require a collection or a pair defining style by analogy, which are not available in our context.

Ma et al. [2014] transfer the style of a mesh to another, using the by-analogy framework to guide an automated copy-paste of geometric patches. This work addresses large scales models while we focus on synthesizing small scale patterns.

Our work Section 4.2 jointly optimizes appearance and structural objectives, instead of transferring style after the facts. The global structure therefore emerges from the details of the pattern, which becomes an intrinsic part of the final shape.

2.3.2 Volume Synthesis

Section 2.3.1 presented methods for synthesizing image textures or geometrical details on surfaces. It is natural to consider extensions to volume synthesis, where the goal is to generate content in 3D volume, called a *solid texture*. The first approaches for solid texture synthesis were based on *procedural textures*, and are discussed briefly in Section 2.3.2. In Section 2.3.2, we focus on by-example methods for solid texture synthesis. Finally, in Section 2.3.3, we briefly discuss techniques that aim to go beyond 3D arrays of voxels, and whose goal is to create more structured content, in a by-example manner. Note that a survey on solid texture synthesis can be found in [Pietroni et al. 2010]. Procedural aspects of volume synthesis for microstructure generation is discussed next in Section 2.4.

Procedural Textures

The idea behind procedural textures, is to define an efficient function, that can be evaluated in constant time and space, at the rendering stage, to display the said texture. *Solid textures* were first introduced by Peachey [1985] and Perlin [1985], as space-filling color functions $\mathcal{F} : \mathbb{R}^3 \rightarrow [0,1]^3$. If $\mathcal{F}$ can be represented compactly, and independently of the resolution at which one evaluates $\mathcal{F}(x)$, then one can produce high-resolution objects at low cost, or at least in a very scalable way. An important class of procedural noise, from which we draw inspiration in Section 5.1, is Worley noise [Worley 1996], which is based on Voronoi diagrams.

The notion of solid texture was extended to *hypertextures*, in [Perlin and Hoffert 1989]. The difference from regular solid texture lies in the function $\mathcal{F} : \mathbb{R}^3 \rightarrow \{0,1\}$,

which now defines the boundary of the object it represents, and not just its color. For further reading on solid textures and hypertextures, the reader is referred to the comprehensive book of Ebert et al. [2003].

Procedural noises can also be defined by means of fractals. In a recent publication, Kim [2015] demonstrated impressive results, where parameters of Julia sets are optimized to fit a prescribed 3D model.

By-Example Solid Textures

The goal of by-example solid texture synthesis is to create a volumetric texture from a set of 2D exemplars [Wei 2002]. In most cases, the input exemplar is simply a triplet of 2D images, one for each axis. Starting from 2D exemplars affords easier artistic control than having to require volumetric data as input. The target domain is generally a 3D array of voxels representing the volume to be filled. Note that for large domains this can become quite prohibitive and this limitation often restricts the use of solid textures in practice. The principle of solid texture synthesis from 2D exemplars is illustrated Figure 2.14.

A first class of algorithms for by-example solid texture synthesis stems from the work of [Jagnow et al. 2004]. In this work, the authors apply techniques from *stereology* to derive a 3D texture from patterns made of particles. The field of stereology is the study of three-dimensional properties of objects from measures observed in two-dimensions. The main drawback of [Jagnow et al. 2004] is that their method requires an explicit modeling of the particle shapes, which has to be done manually. More recently, Du et al. [2013] and Shu et al. [2014] presented other methods specialized to textures made of particles, improving upon the work of Jagnow et al. [2004]. Both techniques remove the limitation of having to model the particle shapes explicitly. Du et al. [2013] reconstruct the 3D shapes of the particles to be distributed in the solid texture, allowing for the synthesis of regular or semi-regular arrangements compared to [Jagnow et al. 2004]. Shu et al. [2014] avoid the expensive step of reconstructing 3D shapes, and represent particles by single points, using a spring-mass system to distribute the sample positions in space.

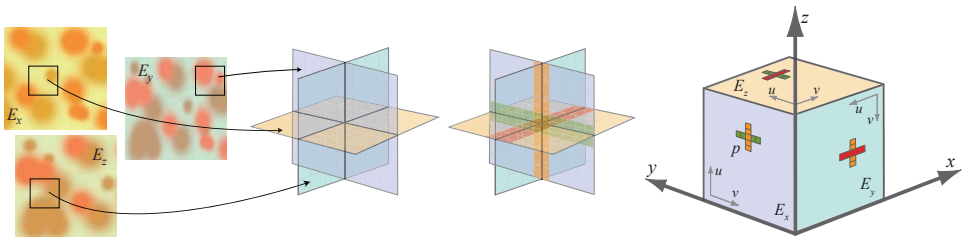
Techniques for more general families of pattern have been also developed. Qin and Yang [2007] presented a framework for the synthesis of gray-scale textures. However, because of the strong correlation of color channels in most texture exemplar, this work in not suited to the synthesis of solid color textures. Kopf et al. [2007] proposed an automated solution for a unstructured color textures, which produced better results than the seminal work of [Wei 2002]. In their work, Kopf et al. perform an energy minimization à la [Kwatra et al. 2005], and rely on orthogonal 2D slices to compute neighborhood similarities. Computing information in a dense voxel grid can be very expensive for large objects. To make the synthesis more efficient, Takayama et al. [2008] proposed to extend the Lapped Texture method from [Praun

et al. 2000] to the synthesis of overlapping patches of textures over a tetrahedral mesh of the volume.

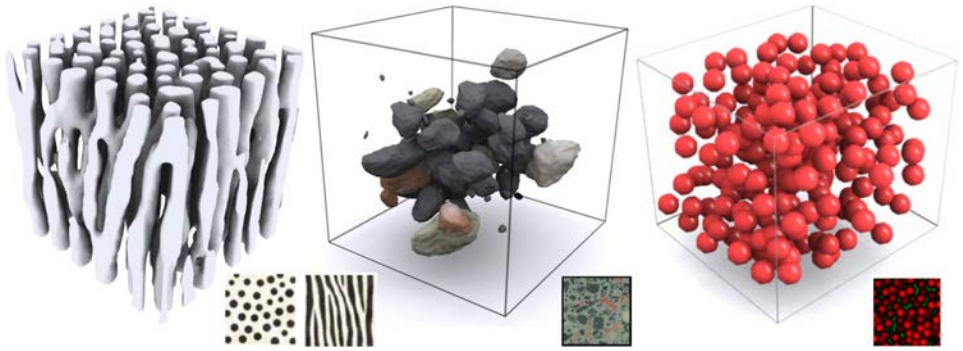
Another approach to accelerate the synthesis in a voxel grid is to perform lazy evaluation as presented in [Dong et al. 2008]. Only the voxels of the visible surface of the model need to be synthesized. This drastically reduces the time complexity of the synthesis compared to [Kopf et al. 2007], affording real time computations. This is another way to achieve surface texture synthesis, and is closely related the synthesis method we present in Section 4.1. Note, however, that the approach of Dong et al. still requires to process a significant amount of voxels, through multiresolution schemes and 3D neighborhoods evaluations. In contrast, our texture synthesis method Section 4.1 requires only a thin voxel layer on the surface, and operates directly on 2D surfacic neighborhoods, without the need for any parameterization of the surface.

Finally, given a solid texture with a distance field channel, Wang et al. [2010] present a method to convert the 3D texture into a vector representation, affording high resolution rendering with a relatively compact storage. This is the solid texture equivalent of vector textures for surfaces [Ray et al. 2005; Nehab and Hoppe 2008].

Figure 2.14 – Solid texture from 2D exemplars [Dong et al. 2008].



(a) Three orthogonal images are used as input (left). The appearance is evaluated by matching neighborhoods from each exemplar across three orthogonal slices in the synthesized volume. Compared to matching full volumetric neighborhood, the computational complexity is much reduced, while preserving the coherence of the result.



(b) Solid texture synthesis results from [Dong et al. 2008].

By-Example Discrete Element Synthesis

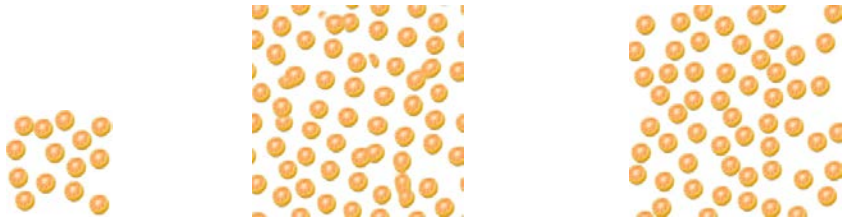
Another way to represent a texture is as a set of distinct elements (e.g. flowers, pebbles) packed together following a certain spatial arrangement, which can be interpreted as a special case of vector-based solid textures. The idea of particle-based texture synthesis was pioneered by Dischler et al. [2002], and in real-time by Lefebvre and Neyret [2003]. Barla et al. [2006] presented a method to synthesis a set of stroke patterns following a Poisson disc distribution. Ijiri et al. [2008] proposed an alternative approach based on local growth of connected patched, allowing more control (e.g. orientation flow) at the expense of some global coherence. Hurtut et al. [2009] improved upon these works by analyzing input exemplars to determine element types automatically, and proposed a multi-type point sampling process to synthesize a distribution of heterogeneous elements similar to a given exemplar. Landes and Soler [2009] further explore the issue of extracting a set of distinct elements from a pixel image. The issue of synthesizing arrangements of discrete elements has been further investigated in [Passos et al. 2010; Öztireli and Gross 2012; AlMeraj et al. 2013a]. In [AlMeraj et al. 2013b], the authors compare to the aforementioned methods by evaluating the perceptual quality of various results.

A more general approach has been proposed in [Ma et al. 2011, 2013], where each element is represented by a set of sample points (in 2D, 3D, and even ND), which allows manipulating elements of various shapes. The technique is illustrated Figure 2.15. The approach is based on an energy minimization technique. The elements can be rigid or deformable, and the neighborhood matching and appearance energy is computed using the point samples only. Landes et al. [2013] fix some of the shortcomings of [Ma et al. 2011] — such as shape interpenetration, or dependence on the initialization — by relying on simplified shape proxies, and use a stochastic process to synthesize the output distribution. Sakurai and Miyata [2014] — although not exactly a by-example approach — proposed a method to model piles of dense aggregate elements, by refining a random distribution of elements on a target shape.

More recently, Roveri et al. [2015] further improve upon the work of [Ma et al. 2011, 2013], by proposing a gradient-based optimization framework that decouples the similarity measure from the sample locations. This leads to a more robust behavior towards bad initial states, and allows the use of a multiresolution scheme to synthesize details of different scales more efficiently. They also use a more robust, two-way, matching metric, which compares positions of samples in both input and output neighborhoods.

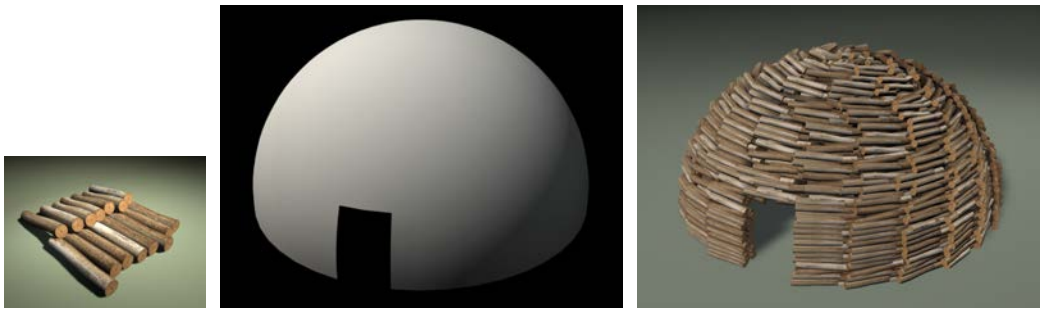
Finally, in the context of fabrication, a recent approach proposed by Schumacher et al. [2016] optimizes the position and the size of a holes on a given surface, and the contours of the holes are described by discrete elements. They combine the point distribution metric from [Ma et al. 2011, 2013] with an evaluation of the mechanical behavior of the resulting object. This is closely related to our work, especially Section 4.3. In Section 4.4 we discuss in more details the differences between this concurrent work and the approaches developed in this thesis.

Figure 2.15 – Texture synthesis with discrete elements [Ma et al. 2011].



(1) Exemplar. **(2)** Pixel-based synthesis. **(3)** Element-based synthesis.

(a) 2D synthesis. **Left:** Input exemplar. **Middle:** Result from standard pixel-based texture synthesis. Note how the elements are not correctly preserved. **Right:** Texture synthesis result using discrete elements. The elements are preserved and their spatial arrangement matches that of the input exemplar.



(b) 3D synthesis. The same algorithm presented in [Ma et al. 2011, 2013] can handle both 2D, 3D, and 4D (videos) synthesis. The logs used as an exemplar (left) provide an easy way for the user to model a wooden house (right).

2.3.3 Structured Content Synthesis

Moving away from synthesis of unstructured data, there has been a progress towards by-example generation of structured content, be it 2D images or 3D models. For instance, Ramanarayanan and Bala [2007] propose an example-based texture synthesis method, guided by a constraint mask that captures the structure of the exemplar image. With a more specialized goal in mind, Lefebvre et al. [2010] synthesize architectural facades by combining seam carving techniques and patch-based texture synthesis methods. More recently, with applications to digital fabrication, Zhou et al. [2014b] generate 1D patterns along curves while precisely controlling their connectivity. While this method can be applied along a first direction $\vec{u}$, and applied a second time along a second, orthogonal, direction $\vec{v}$, to give the effect of a 2D synthesis, it is in practice difficult to extend it to form a true 2D pattern synthesis method. Moreover, connectivity is only one of the geometric constraint presented in Table 2.1, and does not cover structural defects mentioned Section 2.2.1.

Concepts from image texture synthesis inspired new developments in another line of work, called *model synthesis*. The idea of by-example model synthesis was first proposed by Merrell [2007], and was later improved in subsequent works [Merrell and Manocha 2008, 2011]. Their approach relies on constraint-based modeling: the algorithm start from a set of possible states, and greedily assigns values while ensuring that constraints are satisfied. Consequently, the synthesized result is guaranteed to be locally coherent with respect to the input object. While this family of methods work well on architectural models and objects with local similarity, objects with global structures cannot be captured by their local rules.

Repetitive structures is an important part of identifying and synthesizing structured content. Huang et al. [2014b] detect near-regular structures repeated on the surface of 3D models. Duplicating regular structures often provides a more appealing alternative to stretching when sculpting deformable models [Milliez et al. 2013; Rohmer et al. 2015]. Part-based modeling introduces a high number of constraints to ensure consistency of adjacent pieces being built together. Consequently, the underlying combinatorial quickly becomes too expensive, requiring trade-offs, such as restricting the search space for valid configurations [Liu et al. 2015a].

Drawing from different texture synthesis techniques, Lee et al. [2012b] present a method to synthesize new shapes from existing 3D distance field exemplars. This allows retargeting or editing of an existing model, or creating a new object altogether, based on an input exemplar. In the context of fabrication, a recent approach by Kalojanov et al. [2016] uses shape grammars to produce variations of an input shape, in a way that is suitable for manufacturing.

Finally, other recent techniques combining appearance and structured synthesis in the context of digital fabrication have been proposed recently. Chen et al. [2016] and Zehnder et al. [2016] synthesize curve patterns on a surface while considering their structural soundness. Schumacher et al. [2016] distribute elements on a surface and carve out their contour from the input mesh. Note that these works are not only concurrent to ours, but also very similar in spirit to what is presented Chapter 4. Thus, the similarities and differences with our work are discussed at greater lengths Section 4.4.

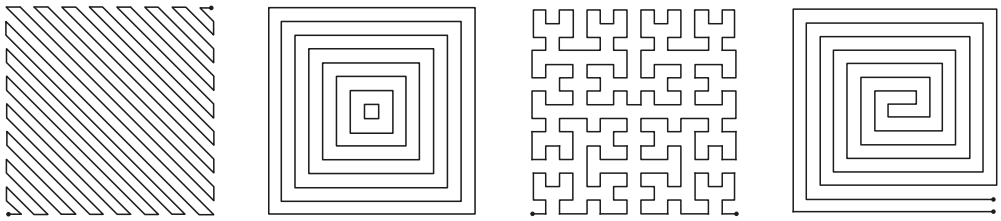
2.4 Infill Patterns and Microstructures

In this section we discuss the synthesis of internal structures for additive manufacturing. When printing a solid object, it is often not desirable to fill the whole volume with the solid material, as it often increases the material cost and print time. Thus, a variety of *infill patterns* are employed to fill the internal volume with more or less dense structures (see Figure 2.16). The goal is usually to find a good compromise for the strength-to-weight ratio, so that the shape does not break once printed, but still saves time and material. Internal structure synthesis can be seen as a very simple form of texture synthesis, where a periodic base tile element is repeated through

the target volume to fill the interior of a shape. However, internal structures do not have to be defined by a periodic pattern: the structures can be optimized globally, or defined via an aperiodic noise function for example, as we will see Chapter 5.

Moreover, if the scale of the pattern is small enough, the *microstructures* thus generated can also affect the average mechanical properties of the solid object as a whole. The equivalent material thusly obtained, from a block of matter filled by a certain microstructure, is called a *meta-material* (as it denotes a new material made from an arrangement of a different base material).

The rest of the section is organized as follows. In Section 2.4.1, we present techniques for synthesizing internal structures in a volume, for the purpose of strengthening an object, or for tiling a prescribed micro-geometry in a efficient manner. In Section 2.4.2, we discuss approaches for modeling and designing objects made of multiple materials. Finally, in Section 2.4.3, we present work related to the design and analysis of the mechanical properties of such microstructures.



(a) Direction parallel. **(b)** Contour parallel. **(c)** Hilbert curve. **(d)** Fermat spiral.

Figure 2.16 – Example of infill patterns that can be used to fill a shape.

2.4.1 Internal Structures

Structural Reinforcement

One of the main objective of infill structures is to minimize the amount of material required to fill a given printed volume, while maximizing the structural strength the physical object has to meet. We note that other objectives are possible. For example, Zhao et al. [2016b] seek to create a connected infill path with low curvature, in order to maximize the print speed and quality on filament printers.

A simple choice of infill pattern with good strength-to-weight ratio is rhombic infill [Lefebvre 2015]. It has the advantage of being easy to print on filament printers, as the overhangs constraints are straightforward to satisfy. Rhombic infills have also received some recent interest in the literature [Wu et al. 2016c; Lee and Lee 2016].

In the computer graphics literature, [Wang et al. 2013] is one of the first work that seeks to maximize the rigidity of an internal structure. The algorithm (see Figure 2.17) optimizes parameters of a truss network, in a manner similar to the topology optimization methods detailed in Section 2.5. The algorithm starts from a

ground structure comprised of beams connecting vertices offseted from the outer surface. Then, a multi-objective scheme is presented, where the goal is to minimize both the volume of the total frame structure, and the number of beams in the final design. A number of constraints are also enforced, so as to prevent excessive elastic deformation or buckling, guarantee minimum thickness, etc. The optimization process operates in two interleaved steps, one that optimizes the topology of the structure (by selecting beams from the ground structure), while the other only changes the geometry (positions and radii $> r_{\min}$) of the selected beams.

Lu et al. [2014] proposed another approach based on distributing Voronoi cells in the volume, and define a honeycomb-like interior structure with closed walls separating the cells. Their algorithm distributes seeds in the target volume, and then iteratively merges neighboring seeds, or updates the amount of hollowing inside individual cells in a stochastic manner. A stress map is computed according to gravity, user-defined loads and attachment points, and is used to drive the hollowing process until a local stress threshold is met. While closed cells are stronger than their open cell counterparts , one disadvantage is that they are not well suited for powder-based printers or stereo-lithography techniques, where the based material needs to escape the interior volume (Figure 2.2d).

Other works have also considered interior volume optimization with structural objectives. Li et al. [2015] use the cross-sectional analysis from [Umetani and Schmidt 2013] to drive the density of a mathematically defined procedural structure inside a 3D model. Zhang et al. [2015c] optimize a frame structure similar the one presented in [Wang et al. 2013]. The main difference is that the algorithm in [Zhang et al. 2015c] is based on a tree structure computed from the medial axis of the shape, so the algorithm can produce results with more internal struts.

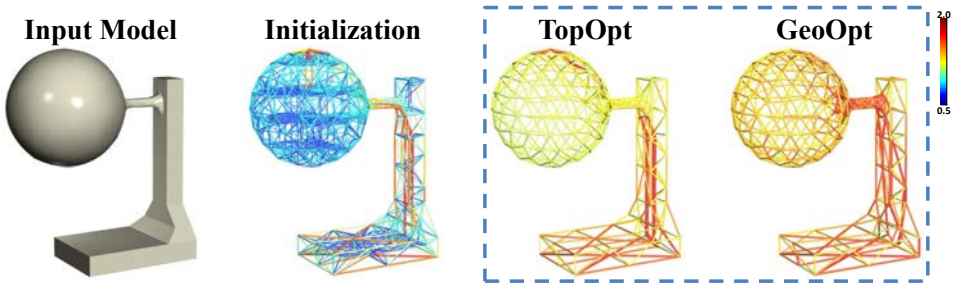


Figure 2.17 – Optimization of internal structures [Wang et al. 2013]. The interior of a given model is filled with beams, whose radii are optimized (potentially removing them), to reduce the material cost while ensuring structural soundness of the printed shape.

Efficient Microstructure Generation

Procedural functions — in the sense of hypertextures defined Section 2.3.2 — where used as early as in 2000 in the context of additive manufacturing, to define the

material composition within a volume [Park et al. 2000]. Chen [2007] use a texture mapping approach to fill a CAD model with a predefined 3D microstructure. This allow for the generation of a variety of patterns by just changing the definition of the base tile.

Pasko et al. [2011] considered procedural definitions of periodic microstructures. The parameters of the microstructures can vary spatially to produce graded materials [Fryazinov et al. 2013], for instance to reinforce an object following a cross-sectional stress analysis [Li et al. 2015]. Brennan-Craddock [2011] computes the intersection of the object and periodic microstructures on a per-slice basis. A frame structure is built on the object surface by subtracting the microstructure cells from a thick surface shell. The surfacic frame structure presented Section 5.1.4 follows a similar intuition for the case of Voronoi foams. OpenFAB [Vidimče et al. 2013] provides a specialized language to describe procedural microstructures. The geometric details are efficiently evaluated at slicing time, streaming voxels to the printer. Similar to these approaches, we evaluate the microstructures procedurally during slicing. However, the microstructures presented Section 5.1 are aperiodic graded foams whose Young's modulus is directly and precisely controlled.

Recent works consider the problem of generating non-periodic microstructures with varying density. Brackett et al. [2014] perform a sequential dithering of a density field to keep a subset of points, which are then used to define an open-cell foam. The adaptive voids approach [Medeiros e Sá et al. 2015] relies on a subdivision scheme to produce denser structures near the surface of an object. Fryazinov et al. [2015] proposed a interpolation scheme based on an interior distance field to vary the parameters of a Voronoi structure inside a 3D model. In [Kou and Tan 2012; You et al. 2016], the authors use B-splines to define 2D porous structures based on Centroidal Voronoi Diagrams. Their method is restricted to 2D, but offers various degrees of control. Yang et al. [2015] consider different families of microstructures defined mathematically, and propose a method to merge and transition between different families in a space partition of the domain. While closely related to our work presented Section 5.1, these approaches do not explicitly control the Young's modulus of the produced structures, nor afford for an efficient parallel evaluation.

Recent software for additive manufacturing propose microstructure generation packages. In particular, *Within* [Autodesk 2016] proposes trabecular structures resembling Voronoi foams. While the parameters can be varied, to the best of the author's knowledge there is no direct control of the Young's modulus and the structures are not defined by a procedure akin to procedural solid textures. The software *Magics* by *Materialise* contains libraries of structures that can be tiled inside a 3D object.

Closer to the procedural solid texture ideology we develop Section 5.1, it is worth mentioning some existing works in the Shadertoy community. The "trabeculum hypertexture" by Neyret (`shadertoy:ltj3Dc`) showcase an example of procedurally defined 3D cellular structures, inspired by [Worley 1996]. While the sampling density is constant, the beam radii is also not precisely controlled. An improved

version in the 2D case was uploaded later on (`shadertoy:MlGGDw`), which provides better control over the beam radii. For the 2D case, the "Voronoi distances" shader by Quílez (`shadertoy:ldl3W8`) demonstrates the correct computation of proper beam radii between adjacent Voronoi cells. However, the 3D extension presents some more tricky cases, which we discuss in Figure 5.3.

2.4.2 Material Design

While the aforementioned infill structures are designed for objects made of a single base material, high-end multimaterial printers allow to mix different materials to grade the properties inside an object. The previously mentioned work of Vidimče et al. [2013] developed a programming framework to describe models made of multiple materials, while Chen et al. [2013] addressed the issue of translating high-level functional specifications into physical properties, thus controlling elastic deformations or reflective properties of a printout. More recently, Ion et al. [2016] and Vidimce et al. [2016] have proposed different systems to help the user interactively design microstructures and create new meta-materials.

To facilitate the design of elastic materials, Li et al. [2014] and Xu et al. [2015b] proposed different algorithms to optimize material properties given an intuitive description of its expected physical response. In a broader context, the notion of example-based material modeling denotes a system where the user would specify just a few keyframe poses of an object, and let an algorithm infer the material properties that matches the deformation described by those keyframes. Techniques for example-based material modeling have gained an increased interest recently, albeit research has been mostly concerned for their applications to computer animation [Martin et al. 2011; Schumacher et al. 2012; Koyama et al. 2012; Zhu et al. 2014; Song et al. 2014; Jones et al. 2014; Zhang et al. 2015b; Jones et al. 2016].

In a more fabrication-oriented context, Chen et al. [2015a] use a data-driven approach to accelerate the FEM simulation of models with a heterogeneous material distribution, and demonstrate their approach on a set of 3D printed objects. Zhang et al. [2016b] present another data-driven approach, for the purpose of controlling the bending behavior of 3D printed models, by optimizing the thickness of their shell. The advantage of this technique is that it is applicable to single material printers, and does not change the exterior appearance of the printed model. In contrast, when microstructures are used to control the elastic properties of a shape, it is difficult to assess how adding a skin shell on the surface, on top of the internal microstructure, will affect the mechanical properties of the printed object [Brennan-Craddock et al. 2012; Aremu et al. 2016].

Finally, in [Xu et al. 2015a], the authors present a method to optimize the material distribution in a tetrahedral mesh, according to a user-specified set of loads and desired displacements. The method then performs dimensionality reduction on the design space by computing the first eigenvectors of a weighting matrix $W = \text{diag}(V_1, \ldots, V_m)$,

where each V_i designates the volume of a tetrahedron. We note that this method bears similarities to topology optimization with density methods — such as the SIMP approach presented in Section 2.5 —, since it optimizes a material distribution to attain a structural objective. However, the main difference is that Xu et al. [2015a] seek to enforce smooth variations over the resulting distribution, while the SIMP method aims at producing binary designs.

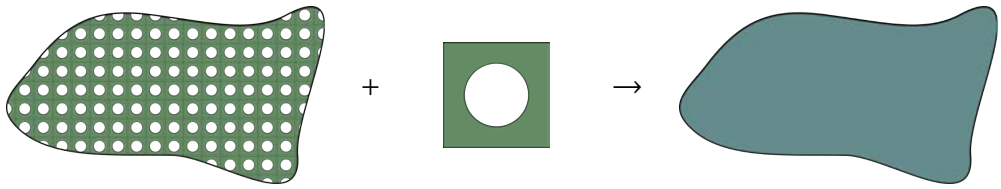
2.4.3 Meta-Materials and Homogenization

Microstructure Reconstruction via Texture Synthesis

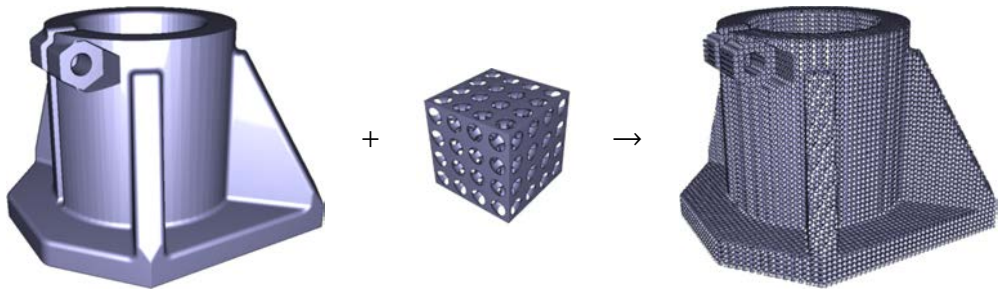
There are different methods to reconstruct a 3D microstructure from 2D cross-sectional information. One of them is solid texture synthesis, and the closely related field of multiple point statistics [Mariethoz and Lefebvre 2014]. In contrast to standard texture synthesis approaches — which are concerned with visual quality and rely on a local neighborhood evaluation —, reconstruction methods for microstructures aim at reproducing statistical properties describing the composite material. The 2D cross-sections are captured experimentally from real samples, and their statistical properties should be preserved in the 3D reconstructed volume. A number of approaches have been proposed to exploit solid texture synthesis methods initially developed for computer graphics (Section 2.3.2), in the context of microstructure reconstruction. Liu and Shapiro [2015] first proposed to use the stochastic optimization process presented in [Wei 2002] to reconstruct 2D and 3D microstructures from 2D samples. Sundararaghavan [2014] applied the more sophisticated approach of Kopf et al. [2007] to the 3D microstructure reconstruction problem, while Turner and Kalidindi [2016] relied on [Chen and Wang 2010], which is basically an improved version of [Kopf et al. 2007]. Taking a different approach, Bostanabad et al. [2016] use a direct supervised learning method to reconstruct volumetric data. Note that all these methods make the common assumption in texture synthesis that the data to reconstruct is unstructured, and can be represented by a local description (neighborhoods matching). Moreover, they do not consider the issue of fabricalibity (Section 2.1.3), which are necessary in our context.

Inverse Homogenization and Functionally Graded Materials

In the field of mechanical engineering Sigmund [1994b, 1995] introduced the optimization of micro-scale structures to achieve specific macro-scale behaviors, such as controlled elasticity. A key idea is to consider the behavior of a composite material, made of infinitely many repetitions of a base material tile. This limit behavior is captured by the theory of *homogenization* [Allaire 2012] which relates the unit material tile to the elastic properties of its (infinite) periodic tiling. This idea is illustrated Figure 2.18. Therefore, most techniques pose an inverse homogenization problem, optimizing for a tile producing a target elastic behavior. Microstructure



(a) When a periodic tile is repeated through a media, the homogenized material corresponds to the equivalent material obtained in the limit case when the size of the representative volume element becomes infinitely small.



(b) The microstructure of a solid object can be represented efficiently when it is defined as a periodic repetition of the same base tile.

Figure 2.18 – Homogenization of a periodic tile (a), and geometry synthesis filling of a solid with a given periodic pattern (b).

optimization via inverse homogenization is investigated in [Sigmund 2000; Sigmund 2009b]. A thorough survey, up to 2013, about microstructure design via inverse homogenization is presented is [Cadman et al. 2013]. Wang et al. [2014b] study the problem under the framework of level-sets. Andreassen et al. [2014] include fabrication constraints in the structure optimization, and present a 3D microstructure with negative Poisson's ratio. Xia and Breitkopf [2015] and Andreassen and Andreasen [2014] present two MATLAB codes for homogenization of periodic structures. Homogenization for additively manufactured structures is discussed in [Liu and Shapiro 2016].

In the field of computer graphics, the design of materials for fabrication has become an important direction of research, as we strive to enable artists and designers to physically realize virtual models of deformable objects [Skouras et al. 2013; Xu et al. 2015a; Pérez et al. 2015]. Bickel et al. [2010] proposed a data-driven approach to design materials with a prescribed deformation. Base materials are stacked by an optimizer to obtain the target properties. Schumacher et al. [2015] extend this idea in two ways. First, elementary material tiles are optimized via inverse homogenization to cover a large spectrum of elastic behaviors. Second, a process globally optimizes for a choice of tiles in a grid covering the object, to achieve the desired, spatially varying elastic behavior. The process considers fabrication constraints and connectivity between adjacent tiles. Panetta et al. [2015] take a different approach, by optimizing for an optimal family of elementary tiles among

a large — but restricted — set of possibilities. The tiles geometry is expressed as an inflated wire mesh, and take into account fabrication and connectivity constraints. Once the best family is determined, it is possible to arrange the tiles in a grid to obtain a spatially varying elasticity.

These works achieve impressive results at relatively low computational costs, thanks to the regularity of the grid and the periodicity assumptions: homogenization is performed on a single base tile instead of having to rely on a global optimization, and the periodicity affords for compact descriptions [Pasko et al. 2011]. However, the tile-based approach has a number of drawbacks.

First, grading the material by changing the tiles within the grid requires careful handling of the interface between neighboring tiles. Ensuring proper transitions between adjacent tiles is essential for the fabrication of functionally graded materials, yet it has received only little attention in mechanical engineering, with notable contributions presented for example in [Zhou and Li 2008; Radman et al. 2013]. In [Panetta et al. 2015], the tiles families are pre-optimized to have matching boundaries. Other methods rely on a global optimization procedure to ensure proper transitions. Alexandersen and Lazarov [2015] optimize directly the microstructures, without homogenization or length-scale separation. In [Schumacher et al. 2015], a global selection process attempts to strike a compromise between continuity across neighbors, grid discretization and elasticity objectives. The global optimization does not scale well with the size of the models — especially as it is desirable to produce the smallest possible tiles to converge towards the limit behavior computed by homogenization. Our approach Section 5.1 circumvent this issue by relying on a direct and simple relationship between elastic properties and geometric parameters.

Second, it is often desirable to conform the elasticity field to the object surface, for instance having a more rigid structure within a distance of the surface. The axis-aligned nature of the grid makes this difficult. One possible way is to optimize for a 3D parameterization of the grid within the object. While this is a topic of intense research (related to hexahedral meshing), this remains difficult [Staten 2007], and the effect on the final elastic properties is hard to precisely establish. Moreover, fabrication constraints, such as islands (Figure 2.2c), would need to be considered through the mapping, making the whole procedure even more complex. Instead, in Section 5.1, we seek to produce *stochastic, aperiodic* structures which are by nature simpler to conform to the gradients of a field as they do not require specific spatial alignments. Finally, the isotropy of periodic tilings is equal to that of the optimized base tile, while the isotropy of stochastic foams can further improve with larger extents of foams (Figure 5.6).

Open-Cell Foams. The procedural microstructures we present Section 5.1 belong to a specific class of microstructures known as *open-cell foams*. These structures occur naturally and can be obtained from physical processes in a variety of materials

including metals. Therefore, in the field of mechanical engineering there has been a strong interest in modeling and analyzing the properties of these structures.

Interestingly, naturally occurring open-cell foams are often idealized as edges of Voronoi cells [Gibson and Ashby 1997]. As we propose to generate foams from procedurally generated Voronoi diagrams, these works are highly relevant for the study of the mechanical behavior of the structures we develop in Section 5.1. In particular, from these studies we can expect the following properties: 1) the elastic behavior of open-cell irregular foams is highly isotropic [Luxner et al. 2007], and 2) their Young's modulus relate *almost linearly* to their geometric parameters (thickness, density) [Gibson and Ashby 1997; Roberts and Garboczi 2002]. This is an exceptionally good property for our purpose of producing graded elastic materials, as the simple relationship affords for a direct derivation of structure parameters given an elasticity target. We verify that our structures meet these observations in Section 5.1.4.

Prior studies also indicate that the Poisson's ratio of open-cell foams does not significantly vary [Gibson and Ashby 1997], and the structures we present in Section 5.1 indeed share this common limitation.

2.5 Topology Optimization

In the previous sections we have presented how problems related to the analysis, editing, or synthesis of 3D shapes for additive manufacturing were approached from a computer graphics perspective. In a broader setting however, shape optimization problems with structural objectives have a long standing history in the mechanical engineering literature, where they have been studied for decades, before 3D became as popular as today. The goal of this section is to provide an introduction to this discipline, known as *topology optimization*, while focusing on challenges relevant in digital fabrication and computer graphics applications. Topology optimization methods seek to answer the very general question: how to distribute matter in a given domain to obtain the best structural performance, without making any a priori assumption on the shape and topology of a possible solution.

The rest of the section is organized as follows. In Section 2.5.1, we present a self-contained, practical introduction to the SIMP approach for topology optimization, as it is used as the foundation for subsequent work in this thesis (Chapter 4 in particular). We also discuss alternative approaches and recent surveys, choices of numerical schemes for gradient-based optimization of structural problems, and point out some practical considerations and available implementations. In Section 2.5.2, we review existing work in topology optimization that specifically considers fabrication constraints for additive manufacturing, such as thickness, overhangs, enclosed voids, etc. Finally, in Section 2.5.3, we review a special category of topology optimization approaches that uses *discrete elements*, i.e. elements with a fixed given shape, as

building blocks for the structure to optimize. The elements are free to move and rotate in the domain. This topic is further explored in our work in Section 4.3.

2.5.1 Introduction to Topology Optimization

Structural optimization problems encompass a broad variety of applications, depending on the choice of materials, physical models, constraints and objectives being considered. In this work, we are primarily interested in solid mechanics problems under linear elasticity, where the goal is to maximize the rigidity of a shape. Nevertheless, we note that techniques for topology optimization can also be applied to a broad range of other physical problems including heat conduction, fluids, electromagnetism, etc.

Regarding the optimization problem itself, one usually distinguishes three categories [Bendsøe and Sigmund 2004]: *sizing* optimization, *shape* optimization, and *topology* optimization, by order of increasing generality.

In *sizing* optimization, such as truss optimization, the structure is given, and the design variables are parameters of the structure (such as the truss radii). By starting from what is called a *ground structure*, and decreasing some radii to 0, it is possible to select which truss should actually be kept in the final design, but it is impossible to add new connections. The most closely related works in the computer graphics literature, that employs truss optimization problems, are probably [Smith et al. 2002; Wang et al. 2013]. Another example of sizing optimization includes [Musialski et al. 2015], where the design variable is the shell thickness of the input mesh.

In contrast, in *shape* optimization problems, a shape is still given as input, but also serves as the design variable in the optimization procedure. In computer graphics, techniques that are based on Laplacian surface deformations, e.g. the surface deformation part of [Prévost et al. 2013] (Section 5), would fall into this category.

Finally, in *topology* optimization problems, the solver is allowed to create new connections, or poke holes in the geometry, thereby changing the topology of the underlying shape. The formulation becomes much more general, and the possible solutions much broader, but the problem is also more difficult to solve. In a sense, the hole-carving part of [Prévost et al. 2013] corresponds to a topology optimization problem. Other computer graphics papers providing recent takes on a more classical version of the topology optimization problem include [Christiansen et al. 2015a; Wu et al. 2016a].

The rest of this subsection is organized as follows. First, we briefly present the major approaches for solving topology optimization problems, and reference recent surveys in the domain. Next, we describe in more detail the SIMP approach, as it needed later in this thesis. We start by formulating the compliance minimization problem, before describing filtering techniques necessary to ensure a proper solution.

Then, we discuss the choice of numerical optimizer for this family of problem. Finally, we present some existing reference implementations available in the literature.

Historical Perspective

The core problem solved by topology optimization is the assignment of a discrete material distribution $\chi(x)$, for each point in the input domain $x \in \Omega$. The characteristic function of the material distribution is denoted by $\chi(x) \in \{0, 1\}$, and evaluates to 0 where there should be no material (void), and to 1 where the solution should be solid. The problem is usually combined with a volume constraint on the solid material, which is expected to be expensive. The goal is to obtain the most efficient structure under a restricted material budget. One thus avoids the trivial solution of filling the domain with solid material — setting $\chi(x) = 1$ everywhere —, which is the solution that maximize the rigidity if one ignores internal body forces (self-weight). In the continuous setting, nothing prevents a theoretical solution from having an infinite number of arbitrarily tiny holes, since it is known to improve the rigidity of the resulting structure [Allaire 2012]. Since designs with infinitesimal holes are impossible to fabricate, it is desirable to impose a length-scale control over the acceptable solution to topology optimization problems, recalling that in practice numerical methods can only solve a discretized version of the problem anyway.

Density Methods. The first practical algorithm for topology optimization is due to Bendsøe and Kikuchi [1988], and relies on a technique known as the homogenization method. A more detailed note on the history of homogenization can be found in the reference book [Allaire 2012]. The idea behind the homogenization method is to optimize the material distribution in a discretized version of the domain Ω, and describe the geometry of each unit cell a small number of parameters (size, orientation, position). By controlling the size/amount of holes in a unit cell, one can affect the density x of the solid material ($x = 1$ means there is no hole). The homogenization method gives the theoretical background to determine the limit mechanical behavior of those unit cells, when the heterogeneous porous media constituted by those holes is replaced by a spatially averaged description : the homogenized composite material.

The homogenization method belongs to the family of density methods, because the material density x is used as one of the optimization variable. Note that to obtain a classical "black-and-white" solution, a penalization needs to be applied, to prevent intermediate densities in the final design. This gave rise to simplified variant of the homogenization method, called the Solid Isotropic Material with Penalization (SIMP) approach, has since then become one of the most popular method for solving topology optimization problems. In contrast to the homogenization method, it discards information about the microstructures geometry, and only retain the material density as the sole variable in the optimization procedure. To encourage a classical $0 - 1$ design without composite media in the result, a penalized density x^p

is used to compute the stiffness of the resulting structure, where $p \geqslant 1$ (a common choice is $p = 3$), hence the name of the method. A consequence of this penalization is that it gives a higher strain energy (compliance) to elements with intermediate densities, which in turns encourages a binary result in the optimized design (see also [Bendsøe and Sigmund 2004]).

The SIMP technique was first suggested in [Bendsøe 1989; Zhou and Rozvany 1991], although the term itself seem to first appear in [Rozvany et al. 1992]. Despite being simpler than the homogenization method, without additional filtering the SIMP formulation is an ill-posed problem: it does not converge to a stable solution when refining the discretization of the domain Ω. Moreover, the lack of physical interpretation behind the penalization process was an obstacle in the adoption of the technique by the community. It was only thanks to the work of Bendsøe and Sigmund [1999] that the SIMP approach became widely accepted in the community. Indeed, Bendsøe and Sigmund [1999] showed that under certain conditions on the penalty exponent p (for example $p \geqslant 3$ in 3D for a base material with a Poisson's ratio of $\frac{1}{3}$), the SIMP model actually falls under the framework of homogenized micro-structural geometries. In other words, for intermediate penalized densities, there exist a model of microstructure geometry that yields the same homogenized stiffness as the penalized stiffness used in the SIMP model. Another fundamental work that contributed to the popularity of the SIMP method is probably the educational paper [Sigmund 2001], where the author present a very simple 99-line MATLAB code implementing the SIMP method. This code can easily be extended to problems other than rigidity/compliance, and has proved to be a good starting point for many people unfamiliar with the domain (including myself).

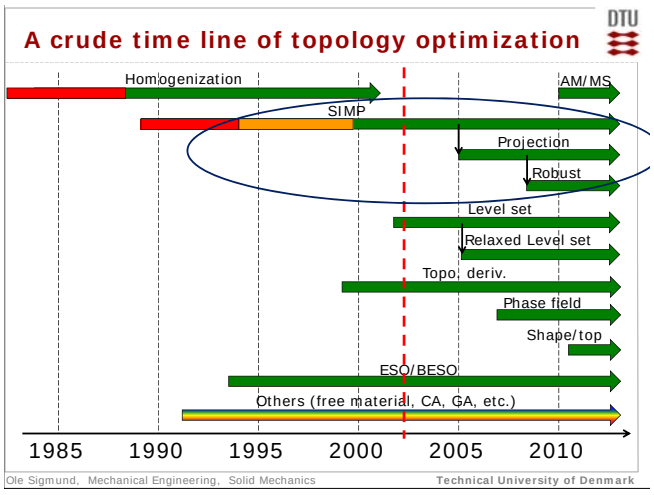


Figure 2.19 – Illustrative timelime of topology optimization methods [Sigmund et al. 2015].

A timeline of the different methods for topology optimization is presented in Figure 2.19. Apart from density methods (homogenization and SIMP), two other important categories are discrete approaches and level-set methods.

Discrete Methods. Discrete approaches cast the density-based formulation into a discrete assignment problem, where the function $\chi(x)$ is assigned a $0-1$ value on a discretized version of the domain Ω. Amongst them, methods such as ESO/BESO employ evolutionary algorithms to update a discrete design. The motivation is to allow exploration of a wider range of configurations, and get closer to a global optimum by avoiding getting stuck around a local minimum, although there is still no strong guarantee that a global optimum will be found. In addition, contrary to the gradient-based optimization used in density approached, discrete approaches usually require a high number of evaluation of the objective function — solving the FEM equilibrium equation —, which can be quite prohibitive, especially in 3D. A critical comparison of discrete approaches, with references to the relevant literature, can be found in the recent survey of Sigmund and Maute [2013]. Evolutionary approaches are also the subject of the last chapter in the book [Allaire 2007].

Level-Set Methods. Instead of using the material density directly as optimization variables, level-set methods (LSM) derive the material densities implicitly from a high-dimensional function Ψ, the level-set function (LSF). The densities are set to 1 where $\Psi(x) < 0$, and to 0 outside, so that the boundary of the structure corresponds to the zero isovalue of the level-set function. The LSF is updated in order to minimize the objective function, and the structure is updated accordingly. Contrary to density methods, level sets provide a better description of the shape boundary — there is no need for density penalization —, and do not suffer from checkerboard issues common in SIMP approaches. However, they do suffer from other drawbacks, in particular regarding a more challenging numerical optimization, imposing the need for regularization techniques, as discussed in the survey [Dijk et al. 2013].

The framework of level sets was originally developed by Osher and Sethian [1988] (see also [Sethian 1999]). The technique has been used for structural optimization problems in [Sethian and Wiegmann 2000; Osher and Santosa 2001], but the formulation the most widely used today has been developed independently by Allaire et al. [2002, 2004] and Wang et al. [2003]. The method combines shape derivative and level sets for front propagation with a "soft" material to describe the void phase. A more detailed history of level-set methods can be found in the recent survey [Dijk et al. 2013], while the older survey of Burger and Osher [2005] provides a more focused discussion on the initial level-set methods.

Because level-set methods are able to merge boundaries, holes in a structure can easily be suppressed, but not created. As a consequence, the optimization procedure is still sensitive to the initial designs, and a shape with multiple holes is typically used as an initial guess. This is a problem especially in 2D, since in 3D the advancing fronts have more freedom — holes can be created by pinching boundaries. To alleviate this issue, a technique commonly employed in combination with level sets is topological derivative. The idea of topological derivatives, developed in 1994 by Eschenauer et al. [1994], is to consider the change in the mechanical response of a shape after poking an infinitesimal hole at a given location. If the change

is beneficial to the structural objective, then a hole is inserted. The technique of topological derivative has been successfully combined with level-set methods in [Burger et al. 2004; Wang et al. 2004; Allaire et al. 2005].

Surveys. Over the past decade, techniques for topology optimization have known a rapid growth (see Figure 2.19). This enthusiasm can be explained by the maturation of the numerical tools being developed and the wide range of industrial applications. More recently, the spread of additive manufacturing technologies contributed to the increased interest towards topology optimization methods, especially in the eyes of the general public. Consequently, an increasing number of surveys are appearing, attempting to cover different important aspects of the field, providing critical reviews of established methods and comparing new trends.

Two excellent books covering the density methods mentioned earlier are [Bendsøe and Sigmund 2004; Allaire 2012]. The first one gives a practical introduction to many different aspects of topology optimization with density methods, while the second one provides a self-contained theoretical framework to the homogenization method. Two major surveys dating back to before the explosion of topology optimization methods in the 2000s are presented in [Eschenauer and Olhoff 2001; Rozvany 2001]. In [Sigmund and Petersson 1998], the authors present a comprehensive survey, discussing the existence of solutions, as well as the different filtering techniques required for solving topology optimization problems using a density approach.

More recently, Rozvany [2009] discusses established methods that are actually used for industrial applications. Sigmund and Maute [2013] present a comparative review of major tendencies in structural optimization, trying to put them in perspective with each others. Deaton and Grandhi [2014] is another recent general survey, with a somewhat larger perspective than [Sigmund and Maute 2013], and also present more applications. [Dijk et al. 2013] contains a comprehensive survey of recent level-set methods, while the older survey [Burger and Osher 2005] offers a more succinct presentation of earlier level-set methods.

An interesting survey comparing and benchmarking different numerical solvers for gradient-based structural optimization using the SIMP method is presented in [Rojas-Labanda and Stolpe 2015b]. More recently, Lazarov et al. [2016] and Liu and Ma [2016] discussed techniques to ensure manufacturability constraints in density-based and level-set-based topology optimization methods, a subject that we discuss in Section 2.5.2. Finally, Sigmund et al. [2016] discuss the optimality of truss-like structures, as opposed to structures comprised of walls with variable thickness.

Problem Formulation

Let us consider the 2D case of compliance minimization, where the goal is to produce the most rigid shape with a limited amount of material. We follow the description from [Sigmund 2001; Andreassen et al. 2011]. The domain Ω is discretized with

a regular grid of square finite elements (Q4). The optimization variables are the element densities $\boldsymbol{x} \in [0,1]^N$, ranging from 0 — representing void "soft" material —, to 1 — representing full "solid" material. The relevant notations are recalled here for simplicity:

$\boldsymbol{x}$	Element density vector
$\mathbf{u}$	Nodal displacement vector
$\mathbf{u}_e$	Local displacement vector for element e
$\mathbf{f}$	External force vector
$\mathbf{K}$	Stiffness matrix
$\mathbf{E}$	Constitutive matrix
E	Young's modulus

Following these notations, the compliance minimization problem can be written as:

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad C(\boldsymbol{x}) = \mathbf{u}^\intercal \mathbf{f} \tag{2.1a}$$

$$\text{subject to} \quad \mathbf{K}\mathbf{u} = \mathbf{f} \tag{2.1b}$$

$$v_{min} \leqslant \sum_e x_e \leqslant v_{max} \tag{2.1c}$$

$$\boldsymbol{x}_{\min} \leqslant \quad \boldsymbol{x} \leqslant \boldsymbol{x}_{\max} \tag{2.1d}$$

The compliance, or strain energy, (2.1a) is the work of the external forces on the system. The more the forces work, the less rigid the shape is. To produce the most rigid shape under these conditions, one must thus seek to minimize the compliance. Equation (2.1c) is a constraint on the volume. Usually v_{min} is set to 0, except e.g. for self-weight problems (see Section 4.2.4). In contrast, it is essential to bound the maximum authorized volume v_{max}. Indeed, in the absence of a constraint v_{max}, a trivial solution would be to fill the domain Ω with solid material, as it is always more rigid than the "soft" void material. Equation (2.1d) is a constraint on the densities. It can be used to represent passive elements (with a fixed density) in the system. Given the equilibrium equation of the system (2.1b), the compliance can be computed as:

$$C(\boldsymbol{x}) = \mathbf{u}^\intercal \mathbf{f} = \mathbf{u}^\intercal \mathbf{K}\mathbf{u} = \sum_e E_e \mathbf{u}_e^\intercal \mathbf{K}_0 \mathbf{u}_e \tag{2.2}$$

The SIMP scheme interpolates E_e, the Young's modulus of element e, as:

$$E_e = E_{min} + x_e^p (E_0 - E_{min}) \tag{2.3}$$

The minimal imposed value of E_{min} ensures that the linear system Equation (2.1b) does not becomes singular, as each square element becomes "infinitely soft". The penalization factor is usually taken $\geqslant 3$ [Bendsøe and Sigmund 1999]. The goal of the penalization is to "push" the variables x towards 0 or 1. Indeed, combined with the limited material budget v_{max}, it is more advantageous to distribute the material following a binary design, as intermediate densities have a "penalized" physical response. We note that other interpolation schemes, such as RAMP, can be used [Stolpe and Svanberg 2001; Bendsøe and Sigmund 2004].

Objective Functions. Other choices of objective functions include stress minimization [Duysinx and Bendsøe 1998; Bruggi and Duysinx 2012; Verbart et al. 2016] (see also §6.10 in the survey [Sigmund and Maute 2013]), optimization of compliant mechanisms [Sigmund 1997; Larsen et al. 1997], inverse shape deformation [Wallin and Ristinmaa 2015]...

Numerical Optimization

To solve the optimization problem posed by eq. (2.1), several methods can be employed. We present the classical approaches based on gradient descent, e.g. via the Optimality Criteria (OC) or the Method of Moving Asymptotes (MMA). Other methods for solving (2.1) can be based for example on interior point methods [Rojas-Labanda and Stolpe 2015a,b]. Note that gradient descent are only able to find a local minimum to a given problem, especially as the system described is (2.1) is non-linear, non-convex, and with non-linear constraints. Very few works are able to guarantee the global minimality of the computed solution, e.g. [Stolpe 2015] in the case of truss optimization.

Solving the Equilibrium Equation (2.1b). One of the key challenge in solving the system (2.1) with a gradient-based approach is that the equilibrium equation (2.1b) needs to be solved every time the objective function needs to be evaluated. Some approaches try to circumvent this issue altogether by solving (2.1b) iteratively along with the design updates, so that the equilibrium is verified only when the final design has been reached. This is known as the Simultaneous Analysis and Design (SAND) approach [Arora and Wang 2005].

In 2D, the linear system (2.1b) is usually small enough that is can be solved using a direct solver on modern computers. For example, CHOLMOD [Chen et al. 2008b] provide a good implementation accelerated on GPU. In practice I have experimented with CHOLMOD through the interface provided by the Eigen library [Guennebaud; Jacob, et al. 2010], which makes it very easy to use.

In 3D however, the system quickly becomes prohibitively expensive. Consider the case of a 64×64×64 grid, which is a relatively small size to represent a complex model. The system has $3 \times 64^3 = 823\,875$ DOFs (not counting fixed nodes). The per-element

stiffness matrix $\mathbf{K}_0$ has 576 nonzero entries for a linear eight-node hexahedron. Assembling the linear system will require iterating over $64^3 \times 576 = 150\,994\,944$ entries, even though the actual global stiffness matrix $\mathbf{K}$ has 60 563 249 nonzero entries in this case (after summation of redundant terms). Using IEEE 754 doubles to store the entries of $\mathbf{K}$ will require 462 MiB of storage, while the densities in the 64^3 grid could fit in 2 MiB. As can be seen, the memory requirement for solving (2.1b) quickly become a huge limiting factor for actual computations. For this reason, the 3D results we show Section 4.2.4 are limited to interleaved planes.

In practice, the linear system (2.1b) is solved using an iterative solver, such as a preconditioned conjugate gradient (PCG), with a geometric multi-grid (GMG) preconditioner. Compared to the related algebraic multi-grid methods (AMG), GMG are better able to take advantage of the specificities of a given problem, since they directly exploit its geometry. In contrast, AMG methods are simpler to implement, because they only need the system matrices as inputs. A GMG preconditioner, in the context of topology optimization, is presented in [Amir et al. 2014]. Regarding AMG preconditioners, I have experimented with the library AmgCL[1], which provides implementations of different AMG preconditioners on the GPU. I was able to easily re-implement the 2D code provided in [Amir et al. 2014] within the framework of AmgCL to solve 3D problems more efficiently. The limiting memory remains an issue though, and thus the performances do not match dedicated solutions such as [Wu et al. 2016a], which are partially matrix-free (they do not store the coefficients of $\mathbf{K}$ on the finest levels of the multi-grid).

Note that, as the solution progresses towards the final binary design, the high contrast between material properties in adjacent elements will cause iterative solvers to converge more slowly. Some works have started to provide contrast-independent solutions, e.g. [Lazarov 2014], but this remains an active area of research. Other precondioning methods, such as domain decompositions — which are most useful for computing a solution in parallel on a cluster of machines —, are beyond the scope of this document.

Updating the Design. In computer graphics, a very popular technique for gradient-based minimization is the BFGS algorithm [Nocedal and Wright 2006]. While it could be used to minimize the system (2.1), there are a number of caveats. First, eq. (2.1) is a system with non-linear constraints, which would need to be taken into account by BFGS, e.g. via an Augmented Lagrangian method [Conn et al. 1991]. Second, BGFS algorithms, and other similar gradient descent methods presented in [Nocedal and Wright 2006], rely on a line search procedure to determine the step size in the descent direction. As we have seen, solving eq. (2.1b) is expensive, so it is desirable to avoid any line search that requires to evaluate the objective function a high number of times.

[1] https://github.com/ddemidov/amgcl

For this reason, structural optimization are solved via different algorithms, which avoid this expensive line search, and have the bonus of being amenable to parallel implementations. The most simple update scheme is the so-called optimality criteria, which can be interpreted as a projected gradient descent (without line search) [Ananiev 2005]. Due to its simplicity and trivially parallel nature, it is a widely popular approach to solve simple structural optimization problems. However, it is a heuristic method which lacks mathematical grounding, and as such is difficult to extend to more complex problems with new objective or constraint functions. For this reason, we preferred to rely on a second class of algorithms, based on Method of Moving Asymptotes (MMA), which is the one we employed Section 4.2.

Historically, the first version MMA was published in [Svanberg 1987]. While still widely used nowadays due to its simpler formulation, it may not always converge to a local minimum from any starting point. A first globally convergent version was published in [Zillober 1993], but it introduced a line search to ensure the convergence of the algorithm. Later, a globally convergent version without line search was presented by Svanberg [1995], but it was too slow in practice. Finally, in a subsequent work, Svanberg proposed another version which was faster and was working well in practice [Svanberg 2002]. This is the version of the Globally Convergent Method of Moving Asymptotes (GCMMA) that is the most widely used nowadays to solve structural optimization problems. A detailed description of both MMA and GCMMA can be found in the technical report [Svanberg 2007], while other variants are presented in [Bruyneel et al. 2002]. Fortran and Matlab code are available upon requests from Pr. Svanberg, while the Matlab version has been recently released[1] under a GPLv3. In practice, we have used the open-source implementation of GCMMA available in NLopt [Johnson 2016], which has been implemented in C independently from the original code of Svanberg.

GCMMA is similar in spirit to Sequential Linear Programming (SLP) methods, but differs in the way it approximate the objective and constraints functions. In [Svanberg 2002], the algorithm is actually presented as part of a more general family called Conservative Convex Separable Approximation (CCSA). The online wiki[2] of NLopt provides a good and succinct description of the MMA/GCMMA algorithms:

> At each point $\mathbf{x}$, MMA forms a local approximation using the gradient of f and the constraint functions, plus a quadratic "penalty" term to make the approximations "conservative" (upper bounds for the exact functions). The precise approximation MMA forms is difficult to describe in a few words, because it includes nonlinear terms consisting of a poles at some distance from $\mathbf{x}$ (outside of the current trust region), almost a kind of Padé approximant. The main point is that the approximation is both convex and separable, making it trivial to solve the approximate optimization by a dual method. Optimizing the approximation leads to a

[1] http://www.smoptit.se/

[2] http://ab-initio.mit.edu/wiki/index.php/NLopt_Algorithms#MMA_.28Method_of_Moving_Asymptotes.2]9_and_CCSA

> new candidate point **x**. The objective and constraints are evaluated at the candidate point. If the approximations were indeed conservative (upper bounds for the actual functions at the candidate point), then the process is restarted at the new **x**. Otherwise, the approximations are made more conservative (by increasing the penalty term) and re-optimized.
> (NLopt online documentation [Johnson 2016].)

Note that MMA stores information about the state variables over three iterations, meaning that $\mathbf{x}^{(i)}$, $\mathbf{x}^{(i-1)}$ and $\mathbf{x}^{(i-2)}$ are stored. This is used to control the evolution of the asymptotes limiting the trust region around the current point. If a variable is oscillating, i.e. $\left(x_e^{(i)} - x_e^{(i-1)}\right)\left(x_e^{(i-1)} - x_e^{(i-2)}\right) < 0$, its asymptotes are brought tighter to each other. Otherwise, the trust region is expanded to speed up the convergence.

Computing the Gradients. Differentiating eq. (2.1a) by element density x_e yields:

$$\frac{\partial C}{\partial x_e} = \frac{\partial \mathbf{u}^\intercal}{\partial x_e}\mathbf{f} + \mathbf{u}^\intercal \frac{\partial \mathbf{f}}{\partial x_e} \tag{2.4}$$

In most cases — e.g. when there is no self-weight, see Section 4.2.4 —, the external forces are constant and do not depend on the densities, hence $\frac{\partial \mathbf{f}}{\partial x_e} = 0$. Differentiating the equilibrium equation (2.1b) by x_e gives the following:

$$\frac{\partial \mathbf{K}}{\partial x_e}\mathbf{u} + \mathbf{K}\frac{\partial \mathbf{u}}{\partial x_e} = \cancel{\frac{\partial \mathbf{f}}{\partial x_e}} = 0 \tag{2.5a}$$

$$\Rightarrow \quad \frac{\partial \mathbf{u}}{\partial x_e} = -\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial x_e}\mathbf{u} \tag{2.5b}$$

$$\Rightarrow \quad \frac{\partial \mathbf{u}^\intercal}{\partial x_e} = -\mathbf{u}^\intercal \frac{\partial \mathbf{K}^\intercal}{\partial x_e}\mathbf{K}^{-\intercal} \tag{2.5c}$$

Now, since the stiffness matrix **K** is symmetric positive definite, we have $\mathbf{K} = \mathbf{K}^\intercal$, and it can deduced that:

$$(2.4) \quad \Rightarrow \quad \frac{\partial C}{\partial x_e} = -\mathbf{u}^\intercal \frac{\partial \mathbf{K}^\intercal}{\partial x_e}\mathbf{K}^{-\intercal}\mathbf{f} \tag{2.6a}$$

$$= -\mathbf{u}^\intercal \frac{\partial \mathbf{K}}{\partial x_e}\mathbf{u} \tag{2.6b}$$

$$= -\frac{\partial E_e}{\partial x_e}\mathbf{u}_e^\intercal \mathbf{K}_0 \mathbf{u}_e \tag{2.6c}$$

This derivation of $\frac{\partial C}{\partial x_e}$ yields a simple formula that does not require the explicit computation of the gradient of the displacements $\frac{\partial \mathbf{u}}{\partial x_e}$. This is possible because the stiffness matrix is symmetric, i.e. $\mathbf{K} = \mathbf{K}^\intercal$. This is known as the self-adjoint method.

Note that because $\mathbf{K}_0$ is symmetric positive definite, $\frac{\partial C}{\partial x_e}$ here is always negative: increasing an element density will always reduce the total compliance of the system. Because the gradient is always $\leqslant 0$, the problem of compliance minimization is a relatively "easy" one, as any heuristic update schemes will probably find a local minimum eventually.

Mesh-Independency Filtering

It has been mentioned before that the compliance minimization problem, as stated in eq. (2.1), is in fact ill-posed, in the sense that a refinement of the finite element grid will lead to different designs: the design does not converge to a stable solution as the grid is refined. This phenomenon is known as the checkerboard pattern, and it has been shown [Díaz and Sigmund 1995; Jog and Haber 1996] to be caused by a bad numerical modeling which overestimates the stiffness of this checkerboard pattern. The problem can be partially alleviated by using higher-order elements, or completely by using a filtering scheme. The checkerboard pattern and filtering scheme is illustrated Figure 2.20.

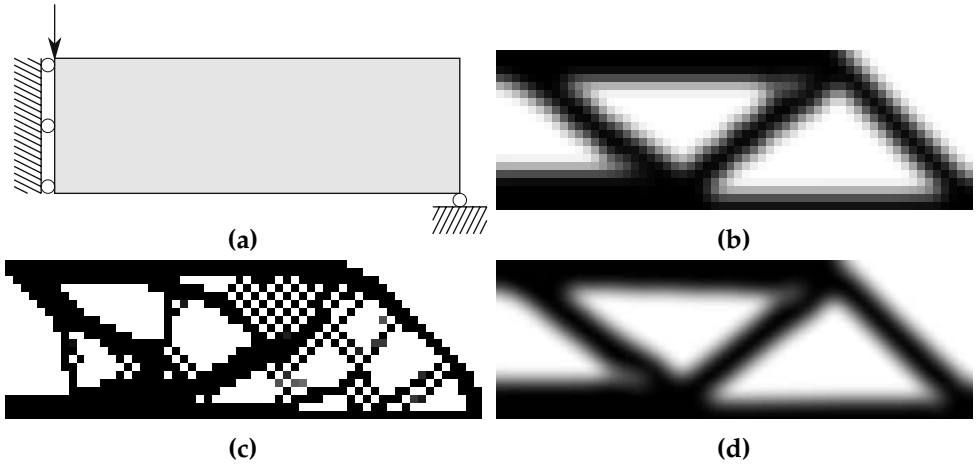


Figure 2.20 – Illustration of the checkerboard problem. (a) Boundary conditions of the classical MBB problem. (b) Low-resolution solution, using a 60×20 grid and a density filter with radius 2.4. (c) Low-resolution solution without filtering. (d) High-resolution solution, using a 300×100 grid and a density filter with radius 12 (which corresponds to 4 % of the domain width).

In this section we present the two most widely used filtering schemes: sensitivity filtering, and density filtering. The idea is to "smooth" the discretized field by a applying a convolution operation to it. If the radius of the convolution is chosen as a

fraction of the domain size, it defines a proper length-scale on design, independent of the discretization refinement. The filtering radius is denoted as $r_{\min}$. Sensitivity filtering is a "hack" that smooths the gradient of the objective function, while the density filtering replaces the physical densities by a smoothed version, and the new gradient is computed via the chain rule.

In the following, the convolution operation is denoted by a matrix $\mathbf{H}$, which is defined as follows:

$$H_{ef} = \max\big(0, r_{\min} - \operatorname{dist}\big(e, f\big)\big) \tag{2.7}$$

In the above formula, $\operatorname{dist}\big(e, f\big)$ is the distance between the centroids of voxels e and f. In addition, we define the column vector $\mathbf{S} = \mathbf{H1}$, whose coefficients are the sum of the columns of $\mathbf{H}$, so that we have $S_e = \sum_f H_{ef}$.

Sensitivity Filtering. This filtering scheme was first introduced in [Sigmund 1994a, 1997]. By replacing the gradient of the objective function with a "smoothed" version, it turns out the checkerboard patterns can be avoided in the final solution. A physical interpretation of this sensitivity filter is given in [Sigmund and Maute 2012].

The gradient of the compliance C is replaced by its filtered version:

$$\widehat{\frac{\partial C}{\partial x_e}} = \frac{1}{\max(\epsilon, x_e) \sum_f H_{ef}} \sum_f H_{ef} x_f \frac{\partial C}{\partial x_f} \tag{2.8}$$

The small value ϵ is there to avoid division by zero, and a typical choice is $\epsilon = 10^{-3}$.

Density Filtering. This second type of filtering scheme was proposed later [Bruns and Tortorelli 2001], and analyzed in more details in [Bourdin 2001]. The physical densities, used to compute the material properties, are replaced by a filtered version $\widetilde{\boldsymbol{x}}(\boldsymbol{x})$, which depends on the optimization variables through the following relation:

$$\widetilde{\boldsymbol{x}} = (\mathbf{H}\boldsymbol{x}) \oslash \mathbf{S} \tag{2.9}$$

Alternatively, the definition eq. (2.9) can be written with element-wise relations:

$$\widetilde{x}_e = \frac{1}{S_e} \sum_f H_{ef} x_f \tag{2.10}$$

The compliance is now a function $C(\widetilde{\boldsymbol{x}}(\boldsymbol{x}))$. Its gradient can be computed via the chain rule as follows:

$$\begin{aligned}\frac{\partial C}{\partial x_e} &= \sum_f \frac{\partial C}{\partial \widetilde{x}_f}\frac{\partial \widetilde{x}_f}{\partial x_e} \\ &= \sum_f \frac{\partial C}{\partial \widetilde{x}_f}\frac{1}{S_f}H_{fe} \\ &= \sum_f H^{\intercal}_{ef}\left(\frac{1}{S_f}\frac{\partial C}{\partial \widetilde{x}_f}\right)\end{aligned} \tag{2.11}$$

The above formula can be expressed more concisely in vector-matrix notation:

$$\frac{\partial C}{\partial \boldsymbol{x}} = \mathbf{H}^{\intercal}\left(\frac{\partial C}{\partial \widetilde{\boldsymbol{x}}} \oslash \mathbf{S}\right) \tag{2.12}$$

Alternative Filtering Schemes. Sensitivity and density filters are two of the most common filtering schemes used in density-based topology optimization, but they are not the only ones. One of the first method for ensuring convergence of the solution with respect to grid refinement was to impose a perimeter constraint, as a way to control the holes appearing in the design [Haber et al. 1996]. A comprehensive survey on filtering schemes and numerical instabilities is presented in [Sigmund and Petersson 1998]. Recent approaches pose the problem of density filtering as a solution to a Helmholtz equation [Lazarov and Sigmund 2011]. This formulation is particularly interesting in the context of parallel computing, because the computation of the solution to the Helmholtz PDE can be split more easily between different nodes. However, in our situation, the solution is computed on a single computer on regular grids. In particular, I have found that implementing eq. (2.9) in an OpenCL kernel on the GPU provided performances good enough for our applications.

More generally, morphological operations such as dilations and erosions can be used to provide different filtering schemes on the densities [Sigmund 2007]. Such formulations have proved a successful tool in optimizing designs that are robust to fabrication constraints and uncertainties in the manufacturing process [Wang et al. 2011]. Interpretations of density filtering and other projection methods in terms of the physical fabrication process (SLA) are given in [Jansen et al. 2013]. Other works that take into account fabrication constraints in topology optimization are presented in Section 2.5.2.

Available Implementations

To demonstrate the practicality and conciseness of implementing topology optimization methods, several works have been proposed, where the authors provide a

complete code accompanied with detailed explanations. Those papers are mostly intended for educational purpose, but they also serve as a starting point for extending classical schemes to more complex situations.

Sigmund [2001] was the first to present a compact implementation of the SIMP method in 99 lines of MATLAB code for solving 2D compliance minimization problems. The code is easy to extend to multiple load case or compliant mechanisms [Bendsøe and Sigmund 2004]. A 3D version[1] is available on the research group webpage of Wang et al. [2005]. The 2D code of Sigmund was later improved and shortened to 88 lines of MATLAB in a subsequent publication [Andreassen et al. 2011], and Python versions of this code can be found on the webpage[2] of their research group. We used this Python code as a starting point to out implementation in Section 4.2. Another 3D code implementing the SIMP method in 169 lines of MATLAB code can be found in [Liu and Tovar 2014]. Finally, ToPy [Hunter 2009] is a Python package that contains a longer, but easily readable code solving compliance minimization, heat conduction and compliant mechanisms design on 2D and 3D grids. The code for ToPy is also available on Github[3].

For optimizing truss structures, a 99-line Mathematica code can be found in [Sokół 2011]. In the case of multi-objective optimization, Suresh [2010] presented a 2D code for efficiently generating Pareto-optimal designs. A 100-line Python code, using an evolutionary approach for compliance minimization instead of a gradient descent, is presented in [Zuo and Xie 2015].

Another implementation of the SIMP method, using unstructured polygonal meshes, is presented in [Talischi et al. 2012]. The algorithm computes solve a FEM system on a polygonal mesh resulting from a Voronoi tessellation. The resulting cells are convex polyhedra whose stiffness matrix can computed by triangulating the cell and evaluating quadrature points inside each of its triangles. The MATLAB code of Talischi et al. [2012] has been ported to a C++ and CUDA in a subsequent publication [Duarte et al. 2015], albeit this new C++ code does not seem to have been made available online.

Regarding efficient code for large-scale simulation, Schmidt and Schulz [2011] describe a C++ and CUDA code for compliance minimization via the SIMP method. Other works describing topology optimization methods on the GPU, but without readily available implementations, can be found in [Wadbro and Berggren 2009; Challis et al. 2014; Wu et al. 2016a]. Amir et al. [2014] provide in supplementary material a 2D MATLAB implementation of a geometric multi-grid preconditioner for efficient solving of the linear system (2.1b). In the context of parallel computing on multiple CPUs, Aage et al. [2015] present an open-source topology optimization framework based on the PETSc library. While achieving impressive and highly detailed results, such a framework is most useful when solving linear systems on

[1] http://ihome.ust.hk/~mywang/download/SIMP3D.m
[2] http://www.topopt.dtu.dk/
[3] https://github.com/williamhunter/topy/wiki

multiple computers, whereas a single desktop computer would still be limited by the available memory.

Implementations for level methods are also available in the literature. On the webpage of Wang et al. [2005], one can find a 199-line MATLAB implementation based on the original idea of Wang et al. [2003] and Allaire et al. [2004]. More recently, a shorter and more educational implementation, featuring only 129 lines of MATLAB code, was presented [Challis 2010]. Liu et al. [2005] described another solution using a level-set method with *FEMLAB*[1]. However, the website where the code is supposed to be downloaded from seems unfortunately unattainable. A boundary variation method — i.e. without changing the topology by merging holes — is described in [Allaire and Pantz 2006]. The implementation is written in `FreeFem++` [Hecht 2012], a high-level modeling language for solving PDEs. Allaire and Pantz [2006] also present an implementation of topology optimization with the homogenization method. More programs can be found on the webpage[2] of their research group.

In order to determine the equivalent material properties of a composite material, Andreassen and Andreasen [2014] presented a simple MATLAB 2D code that performs numerical homogenization. Their implementation is a self-contained, educational code. A 3D extension of this code was used in Section 5.1. For the design of materials with prescribed macroscopic properties, Xia and Breitkopf [2015] describe a self-contained MATLAB code performing inverse homogenization.

2.5.2 Fabrication Constraints

The conception of optimal structures cannot be possible without considering the fabrication constraints associated with industrial manufacturing processes. With the growing interest in additive manufacturing techniques, an increasing research effort is being devoted to the conception of automated algorithms producing shapes that already satisfy different fabrication constraints (some of which are shown Table 2.1).

Note that recent literature surveys providing a more comprehensive overview length-scale control and manufacturing constraints in topology optimization can be found in [Lazarov et al. 2016; Liu and Ma 2016].

Level-set methods are well adapted to control the thickness of a shape. By computing the skeleton (medial axis) of the level-set, it is possible to enforce both minimum and maximum length-scale on the design [Guo et al. 2014b; Michailidis 2014; Allaire et al. 2016]. However, skeletons, especially in 3D, are tricky to compute, because they are very sensitive to noise (see the recent survey [Tagliasacchi et al. 2016]). Wang et al. [2016b] presented another level-set method that achieve thickness control without an explicit skeleton representation, by considering a narrow-band offset of the surface. Using a density-based approach, Guest et al. [2004] present a method

[1]Now renamed *COMSOL Multiphysics* https://www.comsol.com/press/news/article/189/
[2]http://www.cmap.polytechnique.fr/~optopo/index.php

to control the minimum thickness of a structure, by having each cell of the grid influencing its surrounding elements within a prescribed radius. By considering both an eroded and dilated design during the optimization process, Sigmund [2009a] can ensure both minimum thickness and minimum void size. An explicit constraint for the minimum and maximum thickness was proposed in [Zhang et al. 2014], with a technique based on an explicit computation of the image skeleton. Finally, an explicit geometric constraint for minimum length-scale is presented in [Zhou et al. 2015]. The algorithm is inspired by previous level-set methods, but does not resort to computing an explicit skeleton. Instead, a projection scheme is used to rescale the density field in order to satisfy a minimum length-scale constraint. Controlling the minimum void size, Liu et al. [2016] proposed a method to obtain two levels of details on the void phase, corresponding to the rough/finish machining tools.

Beside minimum thickness control, it may also be necessary to account for geometric uncertainties due to manufacturing errors and limited tool precision. Robust formulations considering uncertainties in the fabrication process have been reviewed in [Beyer and Sendhoff 2007; Schuëller and Jensen 2008; Wang et al. 2011], and is still an ongoing topic of research [Schevenels et al. 2011; Lazarov et al. 2012a,b; Zhou et al. 2014a]. A parallel between coated structures and infill patterns in FDM printers is made in [Clausen et al. 2015, 2016], to account for the different behavior at the interface (perimeter toolpath) between the infill pattern and the external void.

More specific to filament printers, the overhang constraint and support detection can be incorporated in the optimization process. Brackett et al. [2011] explicitly analyze the cavities and down-facing edges in the structure, and add a weighted penalty to the objective function at every iteration of the design update. Using a less ad-hoc algorithm, Gaynor et al. [2014] present a projection scheme, where the optimization variables are filtered into a printable design that respect the overhang constraint. The filter function operates by averaging the densities in the supporting pixels below a given point, and thresholding the result via a smoothed Heaviside step function. Langelaar [2016a,b] proposed an improved filtering scheme, which more precisely models the overhang constraints. In particular, it discourages pixels with intermediate densities to support parts which are fully solid. The filter works for both 2D and 3D, and uses a smooth and penalized min/max formulation to threshold the densities according to the supporting elements on the layer immediately below. Note that smooth (differentiable) filters are necessary in order to use a gradient-based optimizer such as MMA. It should be noted that choosing the print orientation *a priori* restricts the set of admissible solutions, which is limiting the performances that can be obtained compared to a solution that first optimizes a design without overhang constraints, and then in a post-processing stage seeks to orient the shape and generate supporting structures. This second approach is used for example in [Leary et al. 2014].

Connectivity constraints are considered in [Liu et al. 2015b; Li et al. 2016], with the goal of removing enclosed voids. The process is achieved by treating the void material as a virtual heat source, with the domain borders set as a dissipative

boundary. The virtual temperature becomes lower when the void material is connected to the boundary, which discourages enclosed voids.

Choice of Representation. Level-set methods are often opposed to density methods such as SIMP for topology optimization. Due to the implicit boundary representation, level-set methods offer certain advantages for imposing geometric constraints such as thickness and slope angle, which are essential for additive manufacturing. However, as outlined in [Dijk et al. 2013], level-set methods and density-based topology optimization can also bear strong similarities, especially when the densities computed from the level-set are embedded in a fixed regular grid. Perhaps a better suited distinction would be to consider techniques that explicitly represent the boundary of the structure, which can be optimized either via level-sets [Dapogny 2013], or via an alternating scheme [Christiansen et al. 2014]. Either way, such an explicit representation facilitates the enforcement of geometric constraints, e.g. for manufacturing purposes [Michailidis 2014], or architectural design [Dapogny et al. 2016].

2.5.3 Layout Optimization with Discrete Elements

As explained earlier, in density-based topology optimization methods, material densities are used directly as optimization variables. In level-set methods, the densities are implicitly derived from the level-set function. However, it is equally possible to compute the densities defining the solid phase through an explicit parameterization of certain geometric primitives. For example, one can imagine combining shapes such as rectangles, rounded bars, or spheres, into a more complex structures, by using topology optimization techniques. This approach, in the context of structural optimization, has been described under different names, such as feature-based optimization, or layout optimization with embedded components. The interest in mechanical engineering is straightforward to imagine: the embedded elements can describe solid components that need to be held together (e.g. in the design of a satellite), or void area where existing pieces need to fit. Contrary to standard passive elements, whose size and shape is fixed in advance, embedded components can move around. In certain cases their scale and number can also be used as a variable (e.g. when assembling a truss-like structure).

Recall that the use of discrete elements as a primitive for texture synthesis and artistic control was discussed in Section 2.3.2. In this section, we focus on aspects regarding the optimal conception of structures, as understood in the mechanical engineering literature. This has become a very active research topic in recent years, and many works mentioned here are in fact concurrent to the development of this thesis. Our work in Section 4.3 further explores ideas for the automatic design of structures using discrete elements in 3D.

In a recent survey, Lazarov et al. [2016] distinguish three approaches for density-based topology optimization using discrete elements. 1) The final shape is defined as

the union of existing discrete shapes, which can lead to a non-differentiable problem, e.g. Saxena [2011] compute the union of overlapping masks describing the void phase, and use an evolutionary method to update the population of masks. 2) The original design field itself is defined as the union of discrete elements, e.g. bars or rectangles [Norato et al. 2015]. While this approach restricts the possible attainable designs, especially when the number of primitive does not change, it remains the most widely used strategy in the works cited this section. 3) The original design field is filtered or projected using filters with a compact support defined by the target discrete elements, e.g. [Guest 2015].

Joint Support Optimization (Density Methods). Early methods for the joint optimization of embedded components (position + orientation) and their supporting structure (holding the components together) can be traced back to before 2000. See the survey of Zhang et al. [2011] and references therein. In their overview, Zhang et al. [2011] present an approach that decompose a discrete shape into a finite number of approximating spheres to treat overlaps between elements. Those elements are placed into a grid domain, which is remeshed locally around the embedded shapes. Subsequent works have explored combination of level-set technique (to define the solid/void density of the embedded components), and the SIMP approach for optimizing their support [Zhang et al. 2012; Kang and Wang 2013; Xia et al. 2013]. Those works use an extended finite element method (XFEM) to ensure proper numerical modeling at the interface between the embedded components and the surrounding support. Wang et al. [2014c] use a similar approach to the design of piezoelectric actuators. In [Zhang et al. 2015a], constraints based on the skeleton of the discrete shapes are used to avoid overlaps.

Layout Optimization (Level Sets). Using a different approach, [Chen et al. 2007, 2008a] are able to combine different shapes together through CSG operations. To obtain a differentiable parameterization of the densities when combining the primitives together, the authors rely on the theory of R-functions [Rvachev 1982]. A similar technique is employed in [Cheng et al. 2006; Mei et al. 2008], which furthermore propose a method to insert new elements, inspired by the notion of topological derivatives. Zhou and Wang [2013] regulates the velocity field used to advect the boundary of the level set via a least-square fitting in order to preserve the geometric characteristics of the embedded features. Liu et al. [2014b] described a unified level-set framework using R-functions to encode geometric constraints. Liu and Ma [2015] presented a 3D scheme that also consider machining constraints via least-square regularization. Kang et al. [2016] developed a level-set approach where explicit constraints are computed based on the distances from the zero level-set, to avoid overlaps between components. In [Zhou et al. 2016], a level-set function is defined using linear approximations of the signed distance functions computed on the embedded elements. Gao et al. [2015a] aggregate the geometric primitives with an adaptive differentiable scheme (based on the number of primitives). Constraints

are imposed to explicitly connect different primitives, by adding a term to the elastic potential energy of the system, so that it is minimized when the displacements at nodes from overlapping primitives are equal under the specified external loads.

Recently, a slightly different approach has also emerged, which tries to produce results comprised of aggregated elements only, i.e. without embedding them into an optimized support structure. In [Guo et al. 2014a; Zhang et al. 2016a], each element generates a field with local support in the shape of the element. The expression of the density field for a given element is a complex expression involving the spatial position and angular orientation of each element. The technique has been further developed in [Deng and Chen 2016], which imposes explicit connectivity constraint between certain elements. Finally, Norato et al. [2015] describe a similar approach, using simper shapes (thick bars with rounded endpoints), and also accommodate for overlapping elements.

Projection Methods. In [Saxena 2011], the void regions are represented by taking the union of a set of disks/ellipsoids/rectangles (called material masks), whose position and number are updated using an evolutionary method. Seeking a differentiable alternative to this simple scheme, Wang et al. [2012] map the masks on a fixed grid using a smoothed Heaviside function, making the update scheme amenable to gradient-based minimization.

In [Clausen et al. 2014], flexible void areas are introduced, and a deformation penalty is computed using a least square measure around the void center of mass. In [Ha and Guest 2014; Guest 2015], a projection method is used to place objects at nodes of the FE mesh. The algorithm needs 1 variable per object to indicate whether the feature is "activated" or not, and a region of "negative influence" around each element discourages phase mixing and overlaps. This method can only place elements are predefined location, and without additional attention suffers from the curse of dimensionality. Finally, Overvelde [2012] use Lagrangian samples inspired by fluid simulation techniques (such as SPH), to define the solid structure. The particles can move through the domain and define the solid implicitly. Our approach Section 4.3 draws inspiration from this Lagrangian formulation, but needs to address different challenges, such as efficient 3D simulation, elements not reduced to a single point, connection between adjacent elements, etc.

Chapter 3

Shape Synthesis with Fabrication Constraints

In this chapter, we are primarily concerned with the question *how to fabricate complex shapes*? In this context, the user provides a 3D shape, and chooses a target manufacturing technology. Because each manufacturing technology has its own set of constraints (discussed Section 2.1.3), the model cannot always be printed as is. In some cases, the shape can still be printed using auxiliary structures, which are removed afterwards. Another point of view is to modify the model beforehand so that it complies with the constraints, e.g. by deforming or cutting the model.

There are several ways a shape can be adapted to fit a specific printing technology. One could classify them as *shape-altering*, and *shape-preserving* algorithms, whether the procedure is allowed to modify the original shape or not. Note that the slicing process unavoidably introduces an approximation of the original shape at the end, so the shape-altering property refers to rather large and visible changes from the input shape. Most techniques described in Section 2.2.2 would qualify as shape-altering, in the sense that they are producing a new surface, hopefully as close to the original as possible, in order to enforce the printing constraints. This is the case of techniques such as Make It Stand [Prévost et al. 2013] and its variants [Christiansen et al. 2015b], which seek to balance a shape before printing so they can rest in static equilibrium in the desired orientation. Another example is methods to reduce the amount of supports required before printing [Hu et al. 2015], where the output of the algorithm is a new shape, that has been deformed.

On the other side of the spectrum, shape-preserving methods have been discussed in Section 2.1.3. This includes techniques for generating external support structures, either *sparse* [Schmidt and Umetani 2014; Vanek et al. 2014a], or *dense* ([Huang et al. 2009b; Heide 2011]. This notion of shape-preserving algorithms can also apply to inner structures generation, either for explicit structural reinforcements [Wang et al. 2013; Lu et al. 2014], or improving print speed [Zhao et al. 2016b]. Segmentation techniques such as [Luo et al. 2012; Hu et al. 2014; Chen et al. 2015b] fall somewhere in between, in the sense that they do not modify the definition of the printed surface, but the decomposition itself introduces visible seams that affect the appearance of the fabricated results. Finally, some methods employ a mix of shape-altering and

shape-preserving techniques: for example the carving step in [Prévost et al. 2013] can be considered shape-preserving, since the surface definition is unchanged, while the shape-deformation step is of course modifying the surface itself, and thus can be labeled as *shape-altering*.

In the context of this thesis, being able to fabricate complex shapes that obey printing constraints is a first logical step towards the conception of a more elaborate printing pipeline — which aims to synthesize new shapes, blending appearance and mechanical considerations in a controllable manner. Thus, before working on shape synthesis *ex nihilo*, I first focused on the problem of shape synthesis for support structures — either external or internal — as they are often needed in conjunction with filament printers.

The rest of the chapter is organized as follows. In the first section a method for generating support structures for filament printers is presented. This fits in the *shape-preserving* category mentioned above, with the target printing technology being fused filament deposition. This work has been the object of a SIGGRAPH publication in [Dumas et al. 2014], and was done in collaboration with Jean HERGEL and Sylvain LEFEBVRE. The content of the original publication is mostly reproduced in Section 3.1, with additional details about the parts where I was the most involved: the core scaffolding generation algorithm (given a set of points to support), collision detection, and the presentation of alternative formulations that we have explored.

The second section of this chapter describes a fast algorithm for performing morphological operations (dilation) on 2D binary images. Possible extensions to 3D dexel data structures is discussed. An interesting application of this work is presented, for the purpose of finding the minimal printable volume enclosing (or enclosed) in a given shape. This particular application has been described in a publication — see the technical report [Hornus et al. 2015], or the later conference version [Hornus et al. 2016]. Although my main contribution to this publication was the implementation of a fast dilation operator for binary images, the details of the particular algorithm were not described in the publication, but are presented here. More precisely, the dilation algorithm relies on an original interpretation of a Voronoi diagram with parallel seed segments. Interestingly, its time complexity does not increase with the dilation radius — in fact it performs faster with larger radii.

3.1 Bridging the Gap: Automated Steady Scaffoldings for 3D Printing

Remark. *Compared to our original publication [Dumas et al. 2014], details about collision detection and possible improvements are further discussed in Section 3.1.4.*

(a) **(b)** **(c)**

Figure 3.1 – The upper leg of the Poppy robot cannot be 3D printed on low cost FDM printers without support. Our technique automatically generates scaffoldings made of horizontal bridges supported by vertical pillars, shown in purple. The print is shown in the middle and on the right after clean up. Bridges are strong and stable, increasing the print reliability while having a low material usage.

Fused Filament Fabrication (FFF) is the process of 3D printing objects from melted plastic filament. The hot plastic exits a nozzle and fuses with the part just below, adding a layer of material to the object being formed. However, filament can only be deposited *on top* of an existing surface. Therefore, overhangs require a disposable *support structure* to be printed, temporarily supporting the threads of plastic that would otherwise hang in empty space.

Existing techniques for support generation fall into two categories: The first allow for very reliable prints by enclosing the bottom of the object in a dense structure, at the expense of increased material usage and build times. The second generate thin hierarchical structures connecting to the surface in a sparse number of points. This uses less material, at the expense of reliability: the part might become unstable, the structure itself may become difficult to print, the bottom surface quality degrades. The user therefore has to correct the structure and its parameters for each new object.

We propose to exploit the ability of FFF printers to print bridges across gaps. Since bridges are always supported by pillars at their extremities, they are both stronger and more stable than hierarchical tree structures. Our technique first selects the points to support based on overhang and part stability during the entire print process. It then optimizes for a printable scaffolding composed of bridges and vertical pillars, supporting all points. The result is an automated support generation technique using little material while ensuring fine surface quality and stability during the printing process.

3.1.1 Introduction

Fused filament fabrication (FFF) is a popular technique for turning 3D models into real, tangible objects. A hot filament is melted through a heated nozzle and fuses with the part just below, adding a layer of material to the object being formed. Advantages are the low cost of both printers and filaments, the relative ease of use requiring few manual steps before and after printing, and the wide availability of printers which can be bought from a variety of manufacturers, e.g. Ultimaker, RepRapPro, 3D Systems, Stratasys.

A major drawback of the process, however, is that filament can only be deposited *on top* of an existing surface. Therefore, overhangs require a disposable structure to be printed, temporarily supporting the threads of plastic that would otherwise hang in empty space. Most of the existing approaches build very reliable support structures that unfortunately incur a large increase in used material and print time. Other approaches produce small support structures but rely on the user to fix an initial, incomplete automated solution. This trial-and-error process often requires multiple test prints. We discuss previous work Section 2.1.

We design our supports to answer two main criteria: 1) supporting plastic threads wherever necessary, while being easy to remove and 2) minimizing used material, while printing reliably. Material usage and reliability are contradictory goals: as the size of the support is decreased it becomes more delicate to print and mechanically fragile. In particular the weight of the object and the friction of the printing nozzle start to endanger the integrity of the structure.

Support generation techniques focus on the issue of overhangs. This is, however, not the only situation in which support is required. Many objects are printed in a fragile equilibrium, or will have subparts in fragile equilibrium at a stage of the printing process. When support is required this issue propagates throughout the entire support structure: a thin support has to be strong enough to support the weight imbalance of the subparts being printed above it. Our supports ensure that not only all overhangs are properly supported, but also that at all stages of the print process the already printed subparts of the object are maintained in a stable position.

Our structures are inspired from the scaffoldings used in construction. In particular, we exploit bridges — horizontal bars printed between mostly vertical pillars. Such bridges can be printed reliably by FFF printers over gaps of several centimeters. These bridges exhibit good mechanical properties and can support other layers of scaffolding. As illustrated in the inset, our scaffoldings are surprisingly strong — this particular scaffolding is made of the same geometry as the scaffoldings we use to support our prints. Despite bridges that are only two threads wide (0.8 mm) and two layers thick (0.4 mm), it withstands the 83 grams of a coffee cup — the rough equivalent of 35 meters of 1.75 mm ABS plastic filament — most objects use less than 10 meters of filament and are supported by several bridges.

The scaffolding weights 0.5 g. These good mechanical properties led us to a scheme where we only consider the topology of the structure: in all practical cases we encountered and despite printing at the smallest thicknesses ensuring reliability, our bridge structures proved sturdy enough.

Contributions.. Our first contribution, described Section 3.1.3, is a novel way to select points to be supported based not only on overhangs but also on the stability of the printed model throughout the entire build process. Our second contribution, described Section 3.1.4, is an efficient algorithm to build bridge scaffoldings that support a given sets of points in space. The structure itself preserves the balance of the print, while using little material. A major difficulty in computing the structure stems from the constraints ensuring that the result is printable: A bridge may only be printed if it is supported by a pillar in each of its extremities.

3.1.2 Printing Bridges

We discuss in Section 3.1.2 our choice of using vertical pillars for the main structure, and compare it with trees. Section 3.1.2 discusses the printing of bridges on FDM printers. Section 3.1.3 describes the selection of the points required for supporting the printed model. Our selection ensures both that deposited filament is properly supported, but also that the printed parts remain stable at all times, throughout the printing process. Section 3.1.4 introduces our bridge construction algorithm: the generation of the geometry of the support structure once the required support points have been determined. We conclude with results and comparisons in Section 3.1.5.

Analysis

An appealing option when designing support structures is to rely on hierarchical tree-like structures. This is a choice worth considering since the support pillars quickly regroup before reaching the printing bed. Therefore, one can expect less material to be used and a smaller print time.

However, this incurs several difficulties. First, printing slanted pillars is generally less reliable than printing vertical pillars. The slope induces a smaller bonding area between layers and an uneven warping during cool down. The reliability varies upon the room temperature, the plastic filament quality, and the layer height settings. Printing vertical pillars is much less sensitive to these factors. Second, even when the structure prints properly, the supported weight and the forces applied by the print head on the upper levels generate torque on the base pillar(s). This bends the structure, and resulting deformations can lead to print failures. In contrast our scaffoldings are stable by construction: the pillars on each side of the bridges prevent the bending of the structure.

One might expect the tree structure to use less material than a bridge structure, thanks to the hierarchical grouping. But as we now discuss, and perhaps somewhat counter-intuitively, bridge structures are in fact comparable to tree-structures. Let us consider a grid of $2^k \times 2^k$ points spaced evenly by one unit (millimeters) on a same horizontal plane. An example is shown Figure 3.2 for $k = 2$. The grid side length is $2^k - 1$ mm. We consider the case of a tree-structure with pillars at 45 degrees. Grouping all points down to a single pillar requires a total pillar length of $L_t(k) = 2^k(2^{k+1}-2)$. The tree then has a height of $\frac{2^k-1}{\sqrt{2}}$. A bridge structure for the same points at the same height requires a total pillar length of $L_b(k) = 2(2^k-1)(1+\sqrt{2}+2^{k-1})$. It can be seen that for $k >= 3$ the bridge is more efficient than the tree. However, as the height further increases the tree grows a single pillar while the bridge grows four. We consider the height at which a tree-structure and a bridge-structure use the same pillar length as a function of k; that is h such that $L_t(k) + h = L_b(k) + 4h$. For instance, for supporting 16×16 points ($k = 4$) spaced by 1 mm the tree becomes interesting again after 56 mm. By this time we reach the configuration shown Figure 3.2 where the tree fails to print properly. In contrast, the four pillars stabilize the bridge structure, making it more reliable to print at higher heights and able to support significant weights.

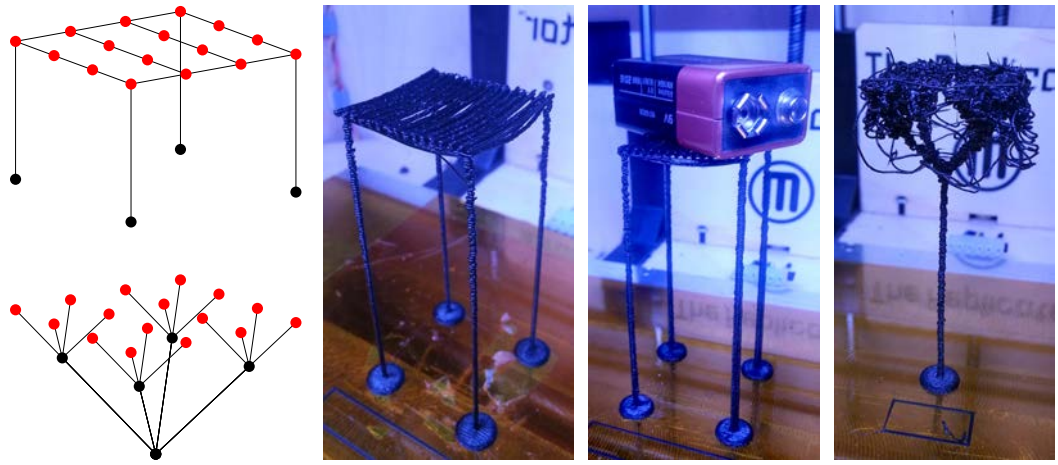
Bridge Printing Reliability

Our bridges are two layers thick and two threads wide, with a small spacing (0.4 mm) in between both threads. To print bridges at relatively high speeds (60 mm/sec) we force the extrusion of a small amount of plastic prior to printing horizontal bridge segments (0.1 mm of filament). Our bridges are designed to print quickly, which negatively impacts their appearance — in particular, the first printed threads often sag. This has however little impact on the quality of the top of the bridge, as detailed in Figure 3.3. While all results shown in the section are printed on a *MakerBot Replicator 1* with ABS plastic, we also tested our bridges successfully on an *Ultimaker 2* with PLA plastic, using same parameters. Our scaffolding algorithm is independent from the exact geometry used to print bridges.

The bridges are mechanically robust despite their small size (see Section 3.1.1). Two 30 mm bridges can for instance support the entire Minotaur model of Figure 3.18 with little deflection (< 0.3 mm).

3.1.3 Support Points Detection

The first step of our approach is to determine a set of points that require support. There are two aspects to this process: supporting filament and ensuring part stability. We start by determining where filament needs support: these points are required regardless of stability, whereas the stability analysis of the overall part depends upon the already placed support points.



(a) (b)

Figure 3.2 – (a) **Top:** A bridge structure supporting a grid of 4×4 points. **Bottom:** Corresponding tree structure. (b) **From left to right:** A bridge structure 78 mm tall, the same structure supporting a 9 V battery, the equivalent tree (same amount of filament). The tree cannot support such weight without toppling. In addition, the top of the tree failed to properly print due to extruder friction. Printed at 60 mm/s on a Replicator 1 Dual with ABS plastic.

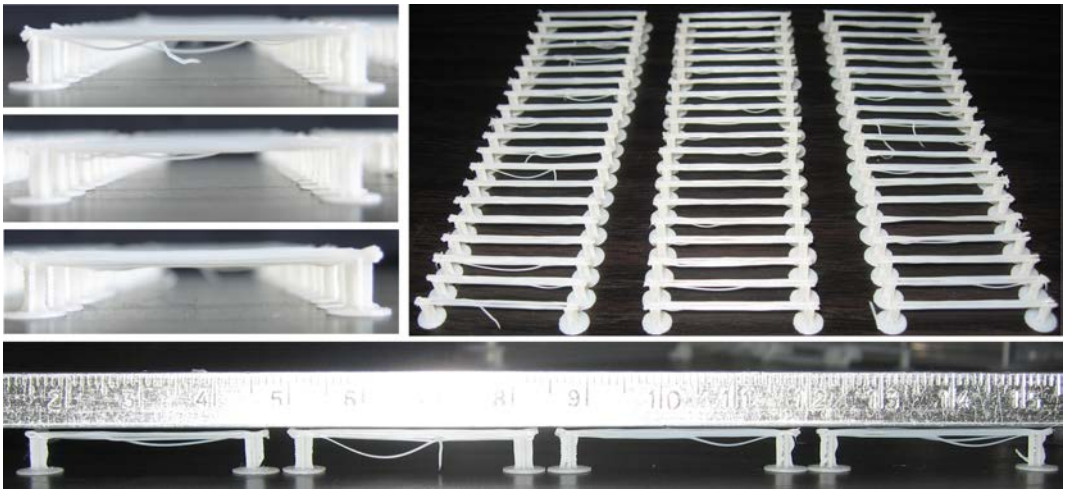


Figure 3.3 – We printed bridges of 30 mm length in batches, along the X, Y axis and at a 45 degree angle. All the bridges printed successfully with a flat top surface. However, the first threads of plastic often sag, resulting in an uneven underside. We measure the deflection to be 0.8 mm on average. In 12 % of the cases a first thread failed to connect and was dangling from the bridge. Printed at 60 mm/s on a *MakerBot Replicator 1* with ABS plastic extruded at 220°C.

Supporting Filament – Detecting Support Points

Our approach starts by slicing the model without any support, determining the full set of *print paths*. Each print path is a sequence of segments along which plastic has to be deposited. The print paths are of two types: the *perimeters* which follow the outer boundary of the surface and the *infills* which fill the interior of the volume.

Our algorithm walks along each print path and checks whether each segment endpoint is properly supported by the layer just below. The test considers the coverage of a disk having for diameter the size of the print nozzle (0.4 mm in our setup). If more than 50 % of the disk lies outside the object on the layer below, we consider the endpoint unsupported — this threshold was determined experimentally to ensure good surface quality. In practice, we implement the test using images of the layers with 0.05 mm pixel resolution. We only check segment end points since the segments themselves form bridges if their extremities are supported. However, in order to ensure a good quality for bottom surfaces, we restrict the longest segment length. This is done by re-sampling any segment whose length exceeds a threshold (5 mm in our implementation).

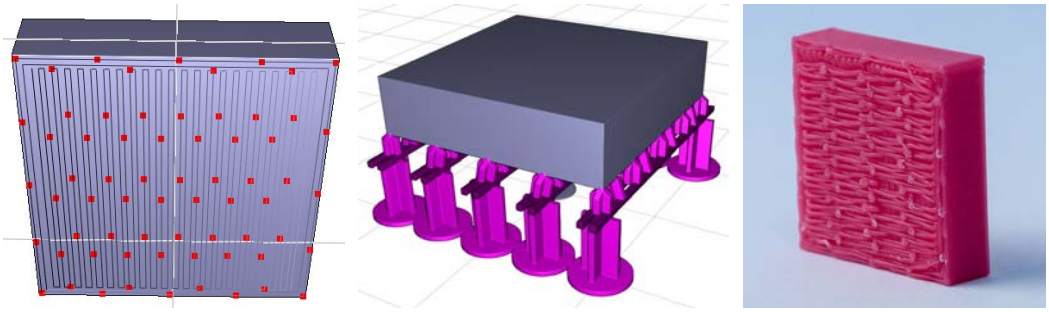


Figure 3.4 – Cube hanging in space. **Left:** Set of points requiring support. **Middle:** Generated scaffold. **Right:** Bottom surface quality.

This simple analysis does not take into account the fact that when a point is supported, a whole length of filament around it can be considered supported as well — the cooling filament having a non negligible rigidity. This is the case on perimeters. This effect turns into a surface support when filaments are deposited side by side, for instance when filling an area with a tight infill pattern (see [Chalasani et al. 1995] for a nomenclature of such cases). We therefore select only a subset of the detected points. Pseudo code is given by Algorithm 1. $C(u, v)$ denotes the curvilinear distance between two points u and v on a same print path; τ is the canceling distance (2 mm in our implementation); IsUnsupported tests the disk coverage test using the layer directly below the point. Figure 3.4 shows the result for a box hanging in space.

Algorithm 1: SELECTSUPPORTPOINTS

Input: Array of paths $\mathcal{P}$, stored as array of points, ordered by Z $\nearrow$.
Output: Set of points $\mathcal{S}$ to be supported.

```
1  foreach perimeter perim ∈ P do
2      foreach u ∈ perim do
3          if IsUnsupported(u) then
4              if ∄v ∈ S , v ∈ perim ∧ C(u, v) < τ then
5                  S ← S ∪ {u}
6  foreach non perimeter path ∈ P do
7      foreach u ∈ path do
8          if IsUnsupported(u) then
9              if ∄v ∈ S , z(v) ⩽ z(u) ∧ ‖u − v‖ < τ then
10                 S ← S ∪ {u}
11 return S
```

Ensuring Part Stability

Once the points required for supporting filament are determined, we consider the stability of the printed part. When printed layer by layer an object made of a single component often starts as disconnected subparts which regroup at higher heights (see Figure 3.5). At an intermediate stage, each of the disconnected sub-parts may be subject to instabilities and topple, leading to a print failure. Our analysis algorithm therefore proceeds in a bottom-up manner, analyzing layer after layer the stability of each subpart independently until they regroup.

We assume next that there is no gluing force between the bed and the printed object — a conservative assumption since any existing gluing force only improves stability. We also assume that the part is rigid when checking for equilibrium and stability.

Stability Conditions. Verifying whether a rigid body lying on a surface is under static equilibrium involves two main concepts: the *center of mass* (CoM) of the object as well as its *base of support* (BoS). For an object of volume Ω of homogeneous density ρ the center of mass is defined as $C_{om}(\Omega) = \rho \int_\Omega \mathbf{x}\, \mathrm{d}x$. In the case of FDM printed object, the volume Ω to consider is not the volume defined by the mesh, but instead the volume resulting from the accumulation of plastic filament — this is especially important when sparse infill patterns are used within the object interior. The BoS is the convex hull of the points of the printed part in contact with the print bed. Our goal is to increase the BoS to ensure stability, through the addition of support points and bridges.

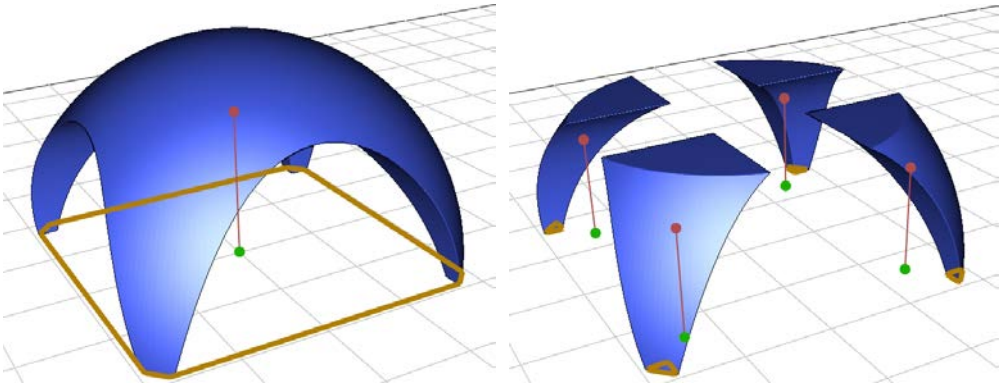


Figure 3.5 – Left: A stable object. **Right:** At an earlier stage of printing the object is made of four disconnected components. The center of mass (red dot) of the full object (left) projects onto the ground (green dot) within the base of support (orange polygon). At an earlier stage of printing (right) the object is made of four disconnected components which are unstable and may topple if unsupported. As can be seen the projected CoM lies outside of the BoS.

A rigid body lies in static equilibrium on a plane orthogonal to gravity if its CoM projects into the BoS (e.g. [Prévost et al. 2013]). This equilibrium might however be unstable if small perturbations — for instance due to the print head — make the object topple. We take these perturbations into account by verifying whether a disk of radius r around the projected CoM lies entirely within the base of support. This is an approximation, and we choose for radius a conservative estimate of 3 mm in our implementation. A more elaborate scheme would change the radius depending of the distribution of mass around the CoM and the distance to the BoS; we did not find this necessary.

When a subpart is detected as unstable we add support points to increase its base of support. This is only possible because the base of support of our scaffoldings always contains all the projected support points: the bridges are always supported by vertical pillars at their extremities.

Algorithm. Our algorithm sweeps through layers from bottom to top. At each layer, we render an image of the print paths (one pixel per 0.05 mm) and use it to keep track of the 3D connected components. Note that this only requires the images of the current layer and the layer directly below. An example of layer image is shown in Figure 3.6, right.

We compute the CoM of each component by summing the coordinates of the pixels belonging to it — each pixel represents a small volume of matter having a same unit mass. Using the image of the print paths properly accounts for infill patterns within the object.

While sweeping through layers we keep track of the current BoS of each 3D component (Figure 3.6, right, green polygons). A BoS is a convex hull and therefore its geometric complexity remains small. Each BoS is initialized at the first layer, where the filament is in direct contact with the print bed: we add all the pixel coordinates belonging to a component to its BoS. At each layer, we include in the BoS the support points detected for the filament. We then check whether the disk of radius r centered on the projected CoM is entirely within the BoS (Figure 3.6, right, orange circles). If that is the case for all components we continue to the next layer.

If the CoM disk leaves its BoS new support points have to be added. During the sweep we keep track of *candidate points* for each component: these are points on the bottom surface of the object that are not projecting in the component BoS. By supporting these points we have an opportunity to enlarge the BoS of the component. We do not consider the candidate point directly but a circle of points around it. The radius is the same as for the CoM disk. This guarantees that we can always enlarge a BoS to cover the CoM disk.

To enlarge a BoS we iteratively add points until the CoM disk is fully covered. At each iteration we add the candidate enlarging the BoS with the largest coverage of the CoM disk. Due to the arbitrary insertion order, some points added in the first iterations may no longer contribute to the final BoS. We remove these and tag the selected points as requiring support. The added points may not be in contact with the surface, in which case we add a small bridge between the candidate point and the surface point (Figure 3.6, right, purple segments). This bridge is given as an input to the scaffolding algorithm.

Figure 3.6, left illustrates the bridge structure resulting from the stability analysis only. Note in particular how the BoS has been enlarged by adding bridges at the first layer, providing an automated raft feature for small contact surfaces with the bed. Timings for this algorithm are summarized in Table 3.1.

3.1.4 Scaffolding Synthesis Algorithm

We now describe how the geometry of our support structure is generated, given the set of points to be supported. We seek to generate a structure so that:

- the structure is formed by vertical pillars and horizontal bridges,
- all required points are properly supported by vertical pillars,
- the vertical pillars connect either to the print bed, to horizontal bridges located below, or to the object itself.
- all horizontal bridges are supported at their extremities, by vertical pillars, by other bridges, or by connecting to the object.

This last point guarantees that the structure is printable on an FDM printer. We search for a structure using a small amount of plastic while enforcing these constraints.

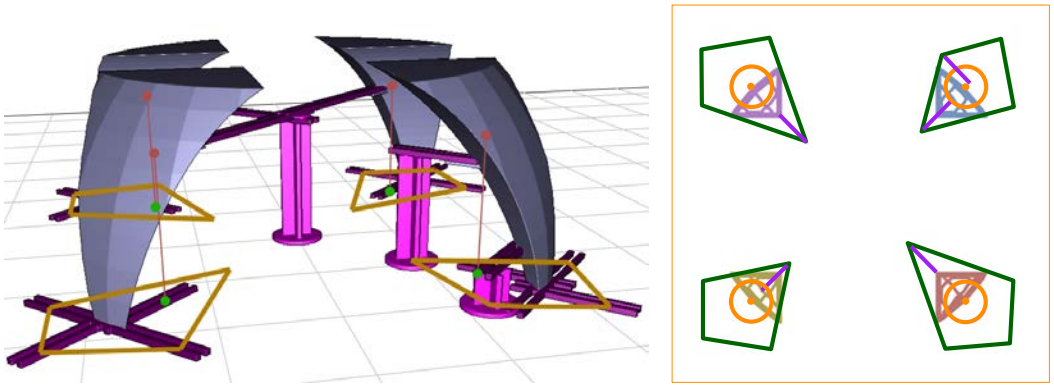


Figure 3.6 – Left: Bridge structure stabilizing the part: the projected CoMs are now within the BoSs. **Right:** Image of a layer created during stability analysis. The print paths are color-coded with the ID of their 3D connected component. The orange circle are the CoM disks, the green polygons the current BoS of each component. The purple segments are the added stabilizing bridges at this stage. They maintain the orange CoM circle within the green BoS polygon.

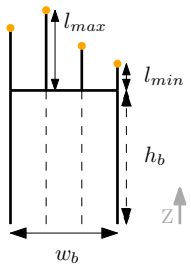
Our initial attempts at formulating an integer problem in a regular grid, capturing the constraints between horizontal bridges and vertical pillars resulted in impractical computation times — it did not terminate but for the simplest examples. This formulation is described in the supplemental material. We therefore propose a greedy optimization algorithm.

We design our algorithm with the ability to create bridges in several directions in the XY plane. This is important since a strong directional bias — such as axis aligned bridges only — would generate a larger number of pillars when supporting features at an angle. We also note that while thin slanted pillars become less reliable as their length increases, they can print reliably on short distances. This is particularly useful when trying to support several points with a rectilinear bridge: perfect alignments are unlikely. Our algorithm therefore has the ability to connect vertical pillars to other elements by adding a small slanted connector at their top.

Bridge Gain and Score

Our algorithm enumerates and selects new bridges that improve the current solution. It therefore requires a function to estimate the benefit of a new bridge. We approximate the bridge benefit by counting the gain and loss in terms of pillar and bridge length.

Following notations in the inset, a bridge of length w_b at height h_b supporting k elements provides a gain of $\mathcal{G}_{ain}(b) = (k-2)h_b - w_b$. Clearly, only bridges supporting more than two points can be beneficial. Our algorithm only inserts bridges where $\mathcal{G}_{ain}(b) > 0$.



When deciding which bridge to insert we compute a score for each bridge. The score is: $\mathcal{S}_{core}(b) = \mathcal{G}_{ain}(b) - k \cdot l_{max}(b)$, where $l_{max}(b)$ is computed as the maximum length of the structure connecting an element above to the bridge. It takes into account non vertical parts that may occur when an element above is not in the vertical plane of the bridge. The score penalizes uneven distributions of connection lengths above the bridge. The bridge giving the best (possibly negative) score will be selected. In cases where the bridge extremities are above the object, we use the free length of each vertical pillar instead of the bridge height: $\mathcal{G}_{ain}(b) = k \cdot h_b - h_1 - h_2 - w_b$ with h_1 and h_2 the heights of the pillars before reaching the object.

l_{min} (see the inset) is a parameter fixing the minimal distance between a bridge and a supported point (1.6 mm in our implementation). Note that lowering the bridge would only reduce its gain. Thus bridges have maximal gain at a distance l_{min} below the lower of the elements they support. This provides a way to efficiently enumerate possible bridge heights.

Construction Algorithm

The input to our bridging algorithm is a set of points and bridges that have to be supported. The bridges come from the stability analysis (Section 3.1.3). We also input a representation of the 3D model allowing for fast line intersection tests — this can be any of the common ray-tracing data-structures.

The output of the algorithm is a set of horizontal bridges and vertical pillars forming a correct (printable) bridge structure while having a small size. Each computed bridge and pillar is turned into actually geometry before 3D printing (Section 3.1.4).

Sweep Strategy. Our algorithm iteratively searches for possible bridges throughout the volume, adding those with the highest score. Since the set of points requiring support is sparse only a small number of possible bridges has to be considered.

Our approach is based on a sweep strategy. Possible bridges are enumerated by sweeping a vertical plane along a fixed direction in the XY-plane. For the sake of clarity let us assume in the following that the selected sweep direction is the X axis. We thus sweep a YZ plane along the X axis, stopping at a number of *events*. The events represent opportunities to create bridges orthogonal to the sweep direction that support other bridges and points located above. The events are the end points and intersections of a number of *anchoring segments* describing where bridges and points can be supported.

Anchoring Segments for Bridges. A bridge is represented by a solid segment supporting elements above. Two anchoring segments — one on each extremity —

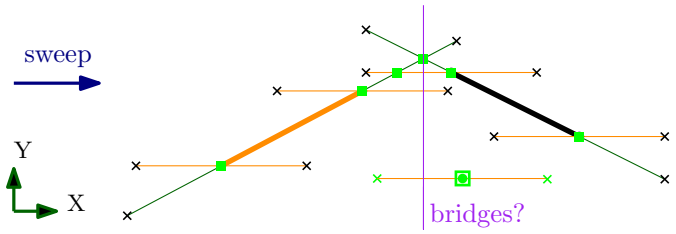
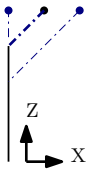


Figure 3.7 – Two bridges and an isolated point as well as their corresponding anchoring segments for a sweep along the X axis. The green squares are events considered during the sweep. The purple line illustrates the YZ sweep plane when examining one event. Bridges will be searched along the Y and Z axis. All intersected anchoring segments are an opportunity for a new bridge to support existing bridges and points.

represent where the bridge can be supported from below: the interval of possible endpoints for this bridge. The anchoring segments for a bridge are outlined in green in Figure 3.7. Their length is chosen to enforce a longest bridge constraint (30 mm in our implementation).

Supporting pillars from below can connect anywhere along the anchoring segments. Once connected, the bridge extends to the pillar and the anchoring segment is removed. The remaining anchoring segment, if any, is updated to enforce the longest bridge constraint. We name this operation a *snap*, and name *open bridge* a bridge with two open anchoring segments and a *half-open bridge* a bridge with a single open anchoring segment. The exact size of the bridge is only determined once both of its anchoring segments are snapped.

Anchoring Segments for Points. Points are supported by pillars grown either from the ground or from a bridge located below. Using only vertical pillars would require a perfect alignment between bridges and points, an unlikely event. Instead, our algorithm allows small slanted bars to be used at the top of vertical pillars, as illustrated in the inset. To ensure printability we constrain the ratio between the offset at the top and the height of the slanted bar. The two dashed lines in the inset show extremal configurations, with the point either exactly above the pillar or at the largest admissible offset (5 mm in our implementation).



Anchoring segments for a point depend on the sweep direction. They are illustrated in orange in Figure 3.7. The anchoring segments for points are created each time a new sweep direction is selected. Note that we consider the extremities of the solid segment of a bridge as points to be supported (Figure 3.7). If one extremity is snapped, the corresponding bridge anchoring segment is removed.

Algorithm 2: GENERATESCAFFOLDING

Input: A set of required points $\mathcal{R} \in \mathbb{R}^3$, a set of required bridges $\mathcal{B}$, the number of sweep directions d.
Output: A valid bridge structure.

```
1  Initialize the active set of elements with E ← R ∪ B;
2  while true do
3      bestBridge ← ∅;
4      for i ← 0 to d − 1 do
5          S ← CreateAnchoringSegments(E, i);
6          P ← ∅ ; // set of segments crossing the current sweeping plane
7          Q ← Events(S) ; // queue sorted by X ↗
8          while Q ≠ ∅ do
9              e ← Pop(Q) ; // leftmost event
10             for s ∈ S starting in e do P ← P ∪ s ;
11             selected ← selectBridge(P);
12             if score(selected) > score(bestBridge) then
13                 bestBridge ← selected
14             for s ∈ S ending in e do P ← P \ s ;
15     if bestBridge = ∅ then return;
16     Let C bet the set of elements supported by bestBridge;
17     for c ∈ C do
18         Snap(c, bestBridge);
19     E ← E ∪ B;
```

Complete Algorithm. The full pseudo-code is given in Algorithm 2. At every iteration (line 2) the algorithm starts with a number of (half-)open bridges and points and attempts to snap as many as possible by adding a new bridge supporting them from below through pillars. When the algorithm cannot find beneficial bridges ($\mathcal{G}(b) > 0$) it stops (line 15).

The algorithm considers multiple sweep directions at each iteration (line 4). Let us assume that the problem is rotated each time to align the sweep direction with the X axis. Possible bridges are enumerated by sweeping the YZ plane along the X axis, stopping at each *event* (lines 7–8). The events are the end points and intersections of all anchoring segments — the intersection being computed after projection on the XY plane, i.e. ignoring the Z coordinate. Figure 3.7 illustrates the set of events for a simple case.

CREATEANCHORINGSEGMENTS creates all anchoring segments for bridges and points (see Figure 3.7); EVENTS computes all events for the sweep (green squares in Figure 3.7). At each event, we consider the bridges that can be formed along the Y axis, connecting the segments currently intersected by the sweep plane. Bridges can

exist at different heights: we consider the heights just below each of the anchoring segments (see Section 3.1.4). This is performed by SELECTBRIDGE (line 11), described in Algorithm 3.

In Algorithm 3, each bridge snapping more than two points is evaluated by the functions ADDSUPPORTED and EVALBRIDGE which implement the bridge gain and score evaluation described in Section 3.1.4. The gain and score are computed assuming that the current endpoints are supported by straight pillars — which is an approximation of the final solution. The score takes into account the increase in length of the bridges located above that are snapped to the new bridge (see l_{max} in Section 3.1.4). Candidate bridges are tested for collisions by COLLISIONS: this checks whether a collision occurs between the model, the bridge and each connector to the elements above it. This takes into account collisions with already placed bridges. The collision tests can be accelerated thanks to the fixed number of directions and the limited set of points to consider.

Finally, SNAP (line 18, Algorithm 2) snaps the extremity c of an element located above to the newly added bridge B by creating a vertical pillar topped (if necessary) by a small slanted pillar. For instance, that would be the case where the purple line in Figure 3.7 intersects orange segments: a slanted bar is necessary to cover the horizontal gap from the point to the bridge. Note that the distance between the point and the intersection along the anchoring segments constrains the height of the bridge. This is taken into account by $\mathsf{z}(c)$ in Algorithm 3, together with the minimum distance between a bridge and an element, l_{min}.

Note that in some cases the new bridge might be able to support two anchoring segments of a same bridge located above ; i.e. a bridge anchoring segment and a point anchoring segment. This case can be seen for the rightmost segment and the purple sweep line in Figure 3.7. If there is more than one possibility to snap an element to the current bridge, only the one yielding the connector of smallest length is considered.

Complexity. Algorithm 2 has a total time complexity of $O(d \cdot (n+k)n^3 \cdot c \cdot b)$, where d is the number of sweeping directions, n the number of points to support, k the number of intersections between the segments projected on the XY plane, c the maximum time complexity of a collision test, and b the number of newly added bridges during the main loop of Algorithm 2. Note that this is only an upper bound, and that in practice the execution time of the whole process is reasonable (see Table 3.1). Several factors are favorable. First, the iterations for different sweep directions are trivial to parallelize. Second, the number of segments input to SELECTBRIDGE can be much smaller than the total number of element n (not all anchoring segments intersect). Finally, for each newly added bridge, all the points it supports are removed, making subsequent iterations within Algorithm 2 faster.

We measured the execution time of our parallel implementation on an Intel® Core™ i7-4770K @ 3.50GHz with 8 threads, 32 GB RAM and a GeForce GTX Titan. We used

Algorithm 3: SELECTBRIDGE along a given plane

Input: Set of segments P intersecting the sweep plane YZ at current event.

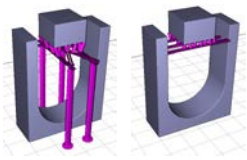
```
1  C ← {P ∩ YZ} ; // intersections sorted by Y ↗
2  Z ← {z(c), c ∈ C} ; // sorted by Z ↗
3  for z ∈ Z do
4      for i1 ← 0 to size(C) − 1 do
5          A ← ∅ ; // points supported by bridge
6          for i2 ← i1 to size(C) − 1 do
7              if ‖C[i2] − C[i1]‖ ⩽ maxDist then
8                  AddSupported(A, C[i2]);
9                  if Collisions(A, i1, i2, z) then break ;
10                 (gain, score) ← EvalBridge(A, i1, i2, z);
11                 if gain > 0 and score > bestscore then
12                     bestbridge ← currentBridge
13 return bestbridge
```

Model	Dim. Max	# Input Pts	# Iter	Bridging	CoM
Knot	45.9 mm	155	4	263 ms	3.9 s
Servojoint	65.6 mm	195	21	676 ms	7.3 s
Gymnast	98.8 mm	114	17	712 ms	19.3 s
Bunny 5cm	55.0 mm	302	42	18 s 283 ms	2.6 s
Hilbert	30.0 mm	262	26	3 s 776 ms	1.2 s
Minotaur	99.5 mm	391	49	43.2 s	13.9 s
Enterprise	152.8 mm	823	75	1 min 12 s	24.2 s
DNA	94.8 mm	867	104	4 min 17 s	16.7 s

Table 3.1 – Execution times on a variety of models.

$d = 8$ sweeping directions (angular increments of $\frac{\pi}{8}$ due to symmetry). As we can see in Table 3.1, the timing for most real-world examples are under five minutes and we believe there is room for further optimizations.

Connecting to the Object. The extremities of a bridge may connect to the top of the surface below by a vertical pillar. As an alternative, we also consider snapping bridges to the object sides, as illustrated in the inset. For each bridge considered in line 10 of Algorithm 3, we also consider extended the same bridge with endpoints until touching the surface (up to the maximum allowable length). Such a candidate bridge simply has one of its $h_{\{1,2\}}$ equals to 0 in the calculation of its score and gain (Section 3.1.4). We retain the candidate bridge with the best score.



Collision Detection

An implementation detail that we have not discussed previously is how to efficiently compute collision queries during the construction of the scaffolding.

Because the scaffolding generation algorithm needs to perform a lot of collision queries during the sweeping process, it is crucial to use an efficient data structure that answers those queries efficiently. Depending on the underlying data structure representing the 3D model, different strategies can be devised. In this section, we briefly explain our implementation for the dexel data structure (Section 2.1.2). Note that a triangle mesh can always be approximated by a dexel data structure in a conservative manner.

There are two types of queries that are being performed: horizontal (in each of the sweep direction), and vertical. The short diagonal pillars are neglected, i.e. we do not test for collision with the original model.

For each of the possible direction of an horizontal query, we build a dexel structure from the corresponding viewing direction. For each layer height we store a binary image of the slice in a compressed format (a 2D dexel array): each row contains a linked list of the entry/exit event in the slice. Using this data structure, it is possible to test if a given horizontal segment $[(x_1, y), (x_2, y)]$ in the image contains a black pixel. The test can be performed in logarithmic time via a binary search. If the segment contains a solid pixel, then it collides with the print and it is rejected.

To account for the thickness of the bridges being generated, a useful trick is to consider the dilation of the original shape by a certain amount, and test if a given line segment intersects with the dilated shape. In practice, we perform a conservative offset in two steps. First, each slice is dilated in 2D by the horizontal width of a bridge. Second, to account for the vertical thickness, we compute the union of all slices within a slab of thickness t around the current one. This union can be computed efficiently via two separate sweeps (one bottom-up and one top-down). Once those two steps are performed, the dexel data structure represent the original shape offset by the right amount (more precisely we have computed the Minkowski sum of the input shape, a vertical segment of length t, and an horizontal disk of radius w).

The storage of a 3D dexel data structure is compact, and only grows linearly with the number of sweep directions used in the scaffolding generation (in practice we only used 8 directions and found it to be sufficient). The important property is that a collision query can be performed in logarithmic time in the number of events along a ray. As most objects have simple topology, the number of such events is usually less than 10, which makes the query very efficient.

For the vertical queries, only one vertical dexel data structure is needed. The same dilation trick can applied, however in this case it is desirable to know precisely

where on the surface a vertical pillar can be attached to, so we did not perform any dilation during the construction of the query data structure.

Similarly to the slanted tips that were neglected before, potential spurious intersections with the original surface are resolved later at the slicing stage by the CSG engine within *IceSL* [Lefebvre 2013]. In practice, a small gap of one or two nozzle widths is enforced between the support structure and the original shape (which "wins" in case of overlaps with the competing supports). The gap makes the support very easy to remove after printing without damaging the surface quality.

Producing the Final Geometry

The final geometry is an union of box and cylinder primitives. We rely on cross-section pillars that we found more reliable to print than cylinders. Bridges are formed of two parallel one-thread thick segments, across two layers. Figure 3.8 reveals the main components of our bridge structures. Our slicer processes union of meshes without suffering from inter-penetration issues.

We adapt our slicer [Lefebvre 2013] for printing the scaffoldings. In each slice we erode by 0.2 mm the bridge structure where it comes in contact with the object. This helps the removal of parts of the structure connecting to the object. For sideways connections to the object (Figure 3.8, second closeup form the left) we add a one-layer thick protrusion from the object, providing a docking space for the bridge. This tiny loop of plastic easily breaks away.

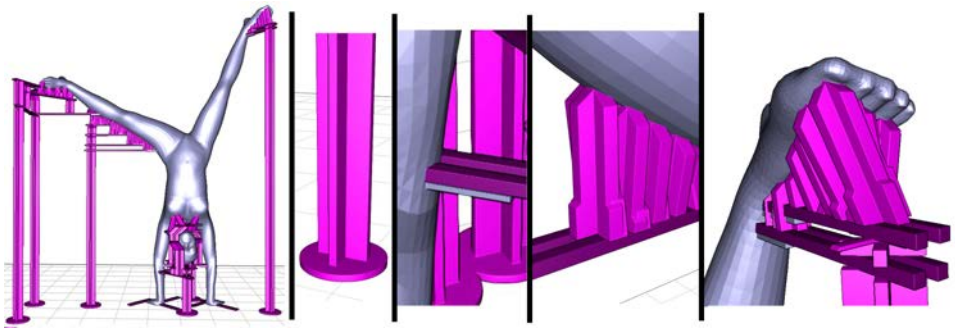


Figure 3.8 – Left to right: The gymnast model and its scaffolding. A pillar, a sideways connector, vertical connectors supporting filament, the mid-air support for the right foot. Model: 123dapp

Possible Improvements

While the support synthesis technique presented in this section generates perfectly valid structures, there are a number of possible improvements that were not discussed in our original publication.

First, even though the construction algorithm presented in Algorithm 2 is quite fast in practice, it still has an asymptotic complexity which is quartic to the number of support points. If the number of input points becomes large, it may be more efficient to reduce it beforehand by doing a first pre-processing pass on the data. Indeed, we often observe that a lot of points are close in space, e.g. under the leg of the gymnast in Figure 3.8. In such cases, a possible approach is to cluster the support points, in order to group the connectors with the same bridge, before trying to connect the bridges together with Algorithm 2. We call the bridge supporting such a group of connectors a *cradle*, and adding this step changes the whole support synthesis pipeline as follows: 1) generate the individual support points, 2) analyze part stability and compute the set of corresponding required bridges, 3) generate all the *cradles* using a fast sweeping method, and 4) generate the final scaffolding supporting the remaining required points and bridges, as defined in Algorithm 2.

A possible heuristic for generating the *cradles* would be to do a first sweep in the Z direction, considering all possible slabs of points within a certain range (corresponding to the maximum height of a connector). Then, a second sweep can be done in the horizontal direction, that tries to fit points in a rectangle whose dimensions correspond to the maximum length of a bridge and to the maximum span of a sideways connector. Then, cradles can be extracted greedily in a first pre-processing step that is in fact a simplified version of Algorithm 2.

While we did not experiment with other heuristics for determining those *cradles*, we did not observe a significant impact on the quality of the final scaffolding. In most cases, the resulting structure would use more plastic than without the pre-processing step, while in a few cases this would result in a gain of material. As most of the examples shown in this section were actually quite small (under 1000 input points), the performances of the bridge synthesis algorithm without any pre-processing to group the points were already satisfying. Consequently, we did not pursue any further development in this direction.

A second improvement that can be discussed concerns the geometry of the connectors themselves. Instead of printing individual pillars to joint each support point to its supporting bridge, it may be advantageous to print a support structure between the bridge and the surface with a single continuous filament path. This idea is illustrated Figure 3.9, and it can be construed as a intermediate between dense supports à la *MakerWare* — for the sine wave joining the surface and the bridge —, and sparse supporting structures. This work has been the object of an internship in our team, and was accomplished in summer 2014 by Florian Abribat, an undergrad student that I helped supervise.

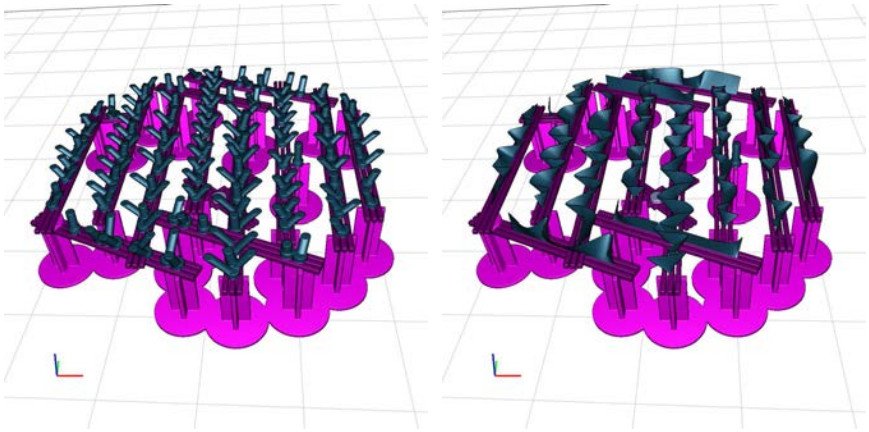


Figure 3.9 – Improvements to the connectors: instead of printing separate connectors to support a surface, one can print a continuous sinusoidal line with a constrained angle.

3.1.5 Results and Discussion

We compare the behavior of different methods running in automatic mode: we perform no manual editing of the proposed support structures. Our technique runs with the exact same settings for all models. All models are printed on a *MakerBot Replicator 1* using ABS plastic at 40 mm/s print speed for perimeters, 60 mm print speed for other print paths (including support) and 120 mm/s travel speed, using layers of 0.2 mm height. The only exception is Table 3.2 where we match the settings of *MakerWare*.

Figure 3.12 compares *MeshMixer* and our technique on the Enterprise 3D model. We generate a support in *MeshMixer* and load it into our slicer to compute the used filament length. We adjusted two parameters in *MeshMixer*: the angle threshold (40) and density (80) to match as closely as possible our support density. *MeshMixer* uses 9.7 m of filament while our technique uses 9.89 m — similar amounts. However, our approach ensures a more uniform support. This is visible under the propulsion units, but also under the main body. On this model, the support from *MakerWare* (not shown) uses 18.8 m of filament as it fills the space below the model.

We show a comparison to *Photoshop CC* in Figure 3.13. With our technique long bridges supported by few pillars replace dense trees. Using parameters from *Photoshop CC* (Replicator 1 profile), *Photoshop CC* uses 4.61 m of filament in the support (11.97 m total) versus 2.31 m for our technique (9.48 m total). Our printed version is shown Figure 3.10.

Figure 3.11 compares our technique and *MeshMixer* on the Minotaur model. The model has heavy overhanging features (arms). *MeshMixer* uses 6.7 m of filament but produces a precarious structure — this is a case where the user would have to manually reinforce the structure through the *MeshMixer* support editing interface.

Figure 3.10 – Top: Printout of the Enterprise model. **Bottom:** Underside after cleanup. We broke the engine connectors during cleanup and had to glue them back (only case where glue was involved). Print time: 3 h 14 min, 9.89 m of filament.

Our technique uses more plastic (7.66 m) but has a denser support, is stable and prints reliably. Our final print and the print from the *MeshMixer* mesh are shown Figure 3.18. *MakerWare*, shown in the inset, uses 10.8 m of plastic.

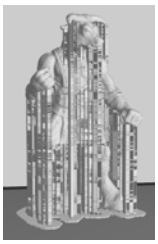


Table 3.2 summarizes other comparisons. In this table we use *MakerWare* as the slicer for the *MeshMixer* output. We use 40 mm/s for printing and 80 mm/s for travel. We match the settings of *MakerWare* in our slicer. *MakerWare* and *MeshMixer* explicitly avoid supporting the bridge overhangs in `Servojoint`. Our technique gives a rougher surface but the bond between layers is improved. Despite the increase in required support points our structure remains three times smaller than the one of *MakerWare*. `Knot` is a defavorable case for our technique. *MakerWare* and *MeshMixer* support very few points through thin beams while our approach builds a full bridge. For this object, with both the *MakerWare* and *MeshMixer* models we had to use a raft for the part to remain stable on the heated bed. In all other cases we use significantly less plastic than *MakerWare*. Our print times are comparable to *MakerWare*, which is in large part due to the printing of the many small connectors. We believe print times could be significantly reduced by grouping connectors supported by a same bridge into continuous connectors.

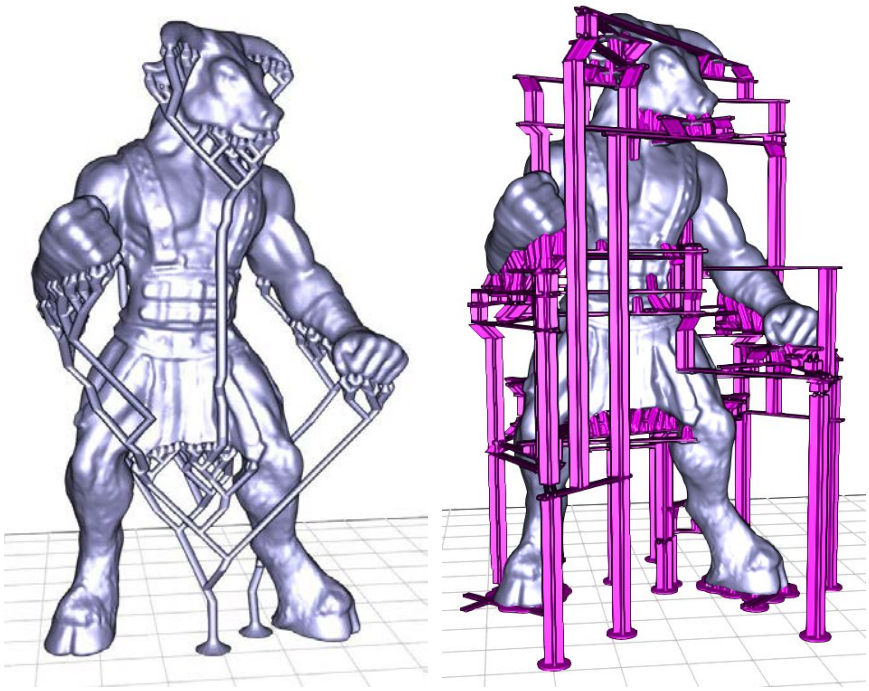


Figure 3.11 – Comparison between *MeshMixer* (left) and our technique (right). *MeshMixer* uses 6.7 m of filament versus 7.66 m for our technique. The tree generated by *MeshMixer* is fragile: a single pillar supports the entire weight of the arms through long slanted bars. Model: `thing:46646` by user ajolivette.

	Servojoint (Figure 3.14)	Knot (Figure 3.15)	Enterprise (Figure 3.10)	Minotaur (Figure 3.18)
MakerWare				
Time	1 h 15	1 h 12	3 h 33	2 h 28
Filament	4.64 m	4.17 m	19 m	10.1 m
MeshMixer ⋆				
Time	1 h 05	1 h 08	3 h 38	2 h 04
Filament	3.44 m	3.85 m	9.7 m	6.7 m
Ours				
Time	1 h 14	1 h 20	3 h 14	2 h 37
Filament	3.86 m	4.01 m	9.89 m	7.66 m

Table 3.2 – Comparison of time and total filament length. Print quality varies, please refer to Figure 3.14 and Figure 3.15.
(⋆) Generated with *MeshMixer* and sliced with *MakerWare*.

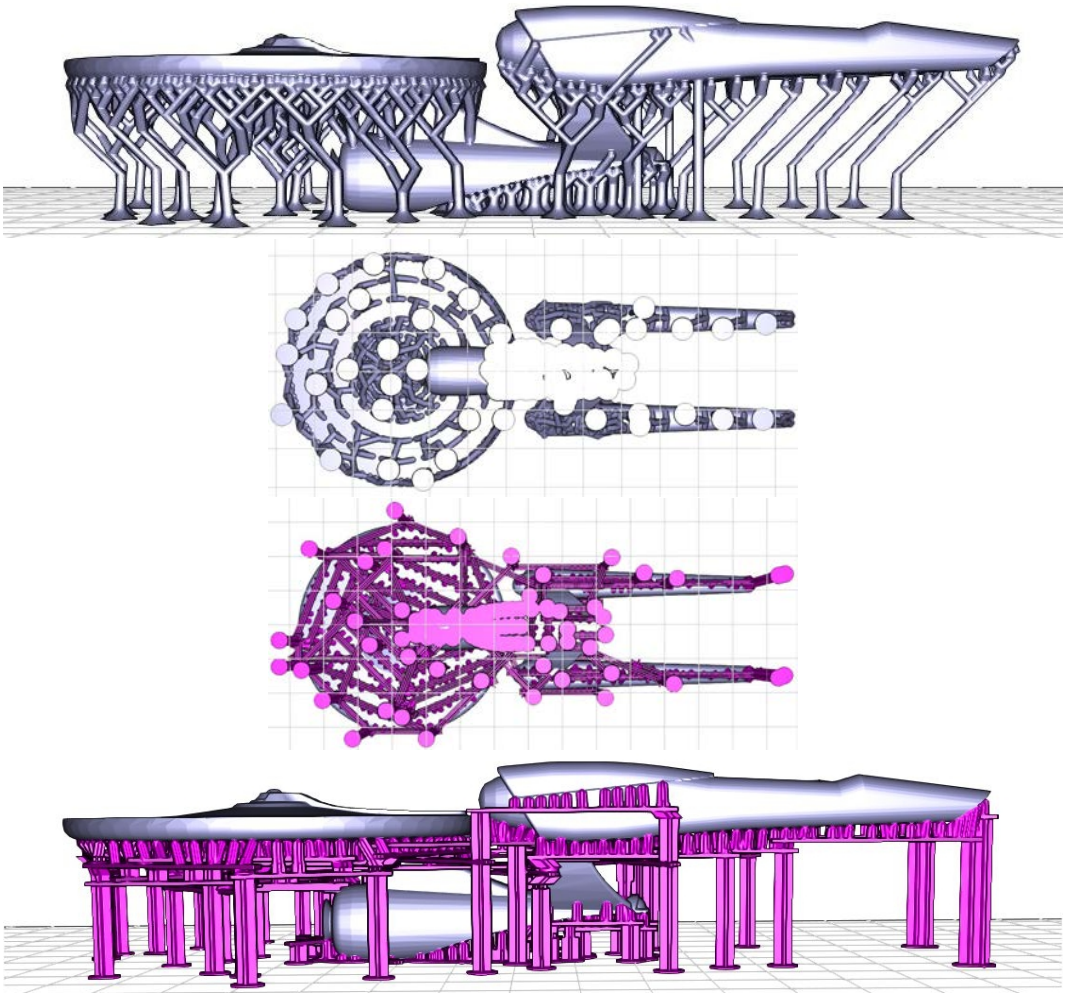


Figure 3.12 – Comparison between *MeshMixer* (top) and our technique (bottom). The middle row shows underside views. *MeshMixer* uses 9.7 m of filament while we use 9.89 m. Notice the much denser support we provide in particular to the rear of the ship.

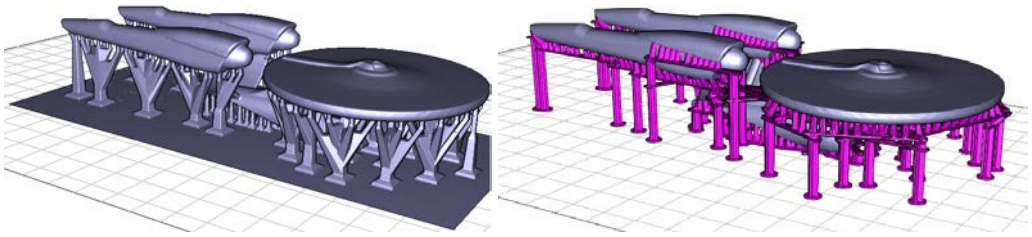


Figure 3.13 – Comparison between *Photoshop CC* (left) and our technique (right). Model: `thing:18346` by user JackSpectre.

Figure 3.14 – Servojoint. *MakerWare*, *MeshMixer*, Ours.

Figure 3.15 – Knot. *MakerWare*, *MeshMixer*, Ours.
Model: `thing:5506` by chylld

Additional Results

Figure 3.16 is a tall and intricate design mimicking the helix of DNA. Figure 3.17 shows three additional models with very different geometries. Note the elegant scaffolding generated for the Gymnast model. The Hilbert cube is a case where bridges have little room to exist — the scaffolding is nevertheless successfully created. Our version of the 5 cm bunny peel model uses 1.97 g of plastic versus only 0.75 g for the *MeshMixer* version, which has been manually optimized (see `http://www.thingiverse.com/thing:131054/#comments`, user meshmixer). Figure 3.1 is printed with PLA plastic on an *Ultimaker 2* with 0.3 mm layer height, all other parameters being the same.

Support Removal

Support detaches from the object when applying gentle force and we clean most objects by hand, sometimes using a small wire cutter on interleaved bridges in complex geometries. The support leaves faint white marks on the plastic — a default shared by all techniques. These can removed by heating the plastic or by finishing the part in acetone vapor. We did not apply such techniques to the results shown here. Soluble material could be used to print our structures.

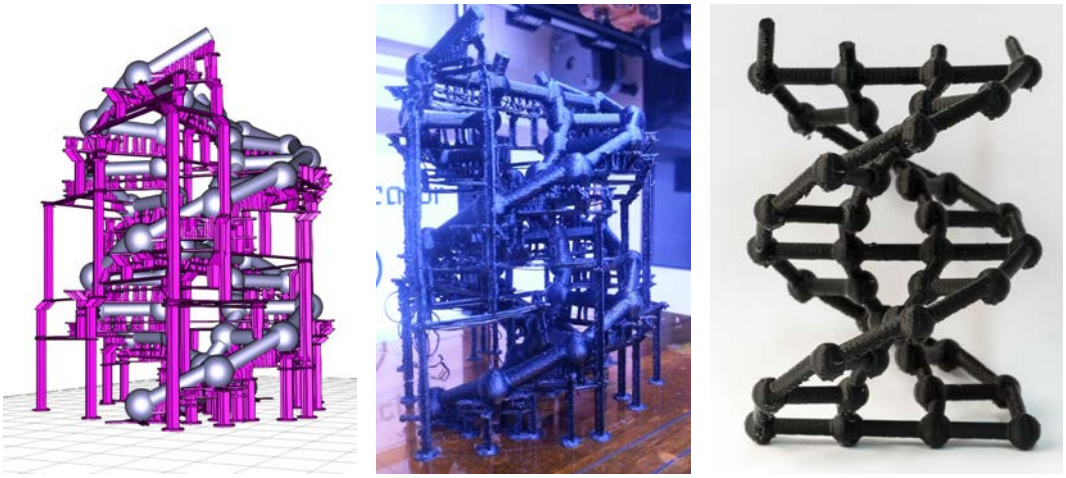


Figure 3.16 – Left: Scaffolding for the DNA model. **Middle:** After printing. **Right:** After cleanup. Printed in 3 h 36 min with 8.7 m of filament.

(a) Gymnast.
Model: 123dapp

(b) Curved Hilbert Cube.
Model: `thing:16343`

(c) 5 cm Bunny Peel.
Model: `thing:131054`

Figure 3.17 – Models printed with our technique, with their scaffoldings (top) and cleaned models (bottom).

Limitations

Our method does not consider the robustness of the printed object. It can be frustrating to discover that a perfect print with support is in fact too fragile — a difficulty worsened by the support removal step. The Enterprise model in Figure 3.10 is such a case: the thin connectors between the engines and the body (< 1.5 mm) broke during clean-up where they connect to the engines. We had to glue them back (no other model required gluing). Automatic methods exist for the purpose of reinforcing objects [Stava et al. 2012].

Our sweeping algorithm is unaware of the geometry of the object aside from the collision detection. For instance, it might miss an opportunity to create a bridge where there is a hole through the object. Analyzing the shape to guide the algorithm is an interesting avenue of future work. This is simple to integrate in our algorithm which already takes open bridges as input.

As explained Section 3.1.2 when printing a bridge the first thread fails in approximately 12 % of cases, leaving hanging filament in the print. This is visible in figures showing the print before cleanup. This has little impact on surface quality as falling filament cools quickly and does not bond with the surface below.

(a) **(b)** **(c)** **(d)**

Figure 3.18 – (a) The Minotaur printed with our technique. Print time: 2 h 37 min, 7.66 m of filament. (b) Attempt at printing the *MeshMixer* version. Print time (same parameters): 2 h 4 min, 6.7 m of filament. We also had to add a raft below each feet. On our model, the stability analysis automatically added a raft beneath each feet (visible in picture). (c-d) Model printed with our technique after clean up.

3.1.6 Alternative Formulation

While searching for an efficient algorithm we also considered alternative formulations. In particular, we considered selecting edges in a regular grid covering the entire build volume. Given a coarse regular grid graph $G = (V, E)$ and a subset of nodes to be supported $\mathcal{R} \subset V$, we seek to select a minimal subset of the edges forming a bridge structure supporting $\mathcal{R}$. Collisions with the part are modeled by removing from E the edges intersecting the object. We model this problem as an integer program that — we hoped — could be solved with off-the-shelves optimizers.

For a node $i \in V$, we denote $\text{left}(i), \text{right}(i), \text{front}(i), \text{back}(i)$ the neighbors of i on the same layer/height, and $\text{above}(i), \text{below}(i)$ the neighbors on the layers above and below. A node i can belong to three types of bridges: horizontal bridges through left(i)-right(i), or through front(i)-back(i), and vertical bridges through above(i)-below(i).

The integer program that we devise uses the following variables:

- x_e equals 1 if the edge $e = \{i, j\} \in E$ is selected in the final support, 0 otherwise.
- u_i equals 1 if the node i *needs to be supported* (either by a bridge or a pillar), 0 otherwise. Initially, $u_i = 1$ for all $i \in \mathcal{R}$.
- b_i^X equals 1 if i is in the *interior* of a horizontal bridge along the X axis, 0 otherwise. b_i^Y is the same for horizontal bridges along the Y axis.

Each edge $e \in E$ is associated a certain cost c_e corresponding to its length. The objective of the linear program is to minimize the total material needed to print the selected edges:

$$\min_{\mathbf{x}} {}^{\intercal} c\mathbf{x}$$

The constraints that express the validity of the support are:

$$x_{i,\text{below}(i)} + b_i^X + b_i^Y \geqslant u_i \qquad \text{for all } i \in V \tag{3.1}$$
$$u_i \geqslant x_{i,\text{above}(i)} \qquad \text{for all } i \in V \tag{3.2}$$
$$x_{i,\text{left}(i)} \geqslant b_i^X \tag{3.3}$$
$$x_{i,\text{right}(i)} \geqslant b_i^X \tag{3.4}$$
$$x_{i,\text{front}(i)} \geqslant b_i^Y \tag{3.5}$$
$$x_{i,\text{back}(i)} \geqslant b_i^Y \tag{3.6}$$
$$x_{i,\text{below}(i)} + b_i^X \geqslant x_{i,\text{left}(i)} \tag{3.7}$$
$$x_{i,\text{below}(i)} + b_i^X \geqslant x_{i,\text{right}(i)} \tag{3.8}$$
$$x_{i,\text{below}(i)} + b_i^Y \geqslant x_{i,\text{front}(i)} \tag{3.9}$$
$$x_{i,\text{below}(i)} + b_i^Y \geqslant x_{i,\text{back}(i)} \tag{3.10}$$

Equation (3.1) states the fact that node $i \in V$ is supported by a pillar below or is in the interior of a bridge. Equation (3.2) denotes the fact that node $i \in V$ needs to be supported (a pillar drawn from above ends up at node i). Equation (3.3) to Equation (3.6) express the validity of the created bridges: if there is an horizontal edge selected with endpoint i, then either i lies on the interior of a bridge with a similar alignment, or some pillar stretches from i to a neighbor below. Finally, Equation (3.7) to Equation (3.10) propagate bridges from left to right and front to back, respectively.

We implemented this integer program in the Gurobi[1] solver. Unfortunately, this approach is impractical as soon as the grid size increases. In addition, compared to our sweeping algorithm, the bridges are constrained by the grid: they cannot follow features at an angle in the XY plane which typically results in more pillars then necessary.

On the simple box model shown Figure 3.19 the optimizer took 20 minutes. We ran the optimizer for more than 24 hours on a model of small complexity ($\approx$ 200 elements to support) without reaching a solution. This motivated our choice of a greedy algorithm.

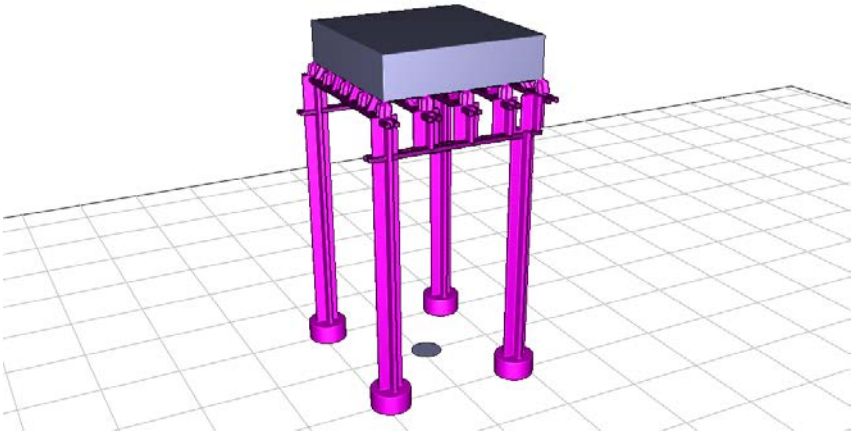


Figure 3.19 – A scaffolding computed by the integer program on a regular grid.

[1] http://www.gurobi.com

3.1.7 Conclusion

We have shown how to exploit a specific property of FFF printers — their ability to print bridges across gaps — to construct reliable scaffoldings. Their geometry gives to our scaffolding interesting mechanical properties that makes them sturdier and more stable, even at the smallest thickness ensuring that they print correctly. Our structures could probably benefit other processes such as stereolithography — but the set of requirements are different.

Further reducing the quantity of material usage while preserving reliability will require a precise modeling of the mechanical properties of the structure and object throughout the print process. This is a challenging task since the plastic deposited in layers has an anisotropic behavior which we expect to become highly nonlinear on thin slanted structures. This is nevertheless an exciting venue of future work. In the meantime our technique provides a simple and reliable way to print interesting and complex geometries with a reasonable material usage.

3.2 Fast Discrete Morphological Operations With Half-Space Voronoi Diagrams

Morphological operations such as dilation and erosion are ubiquitous in all areas of computer science. They can be used e.g. to regularize shapes [Williams and Rossignac 2005], ensure robust designs in topology optimization [Sigmund 2007], compute image skeletons, or to compute support structures in the context of digital fabrication [Hornus et al. 2016].

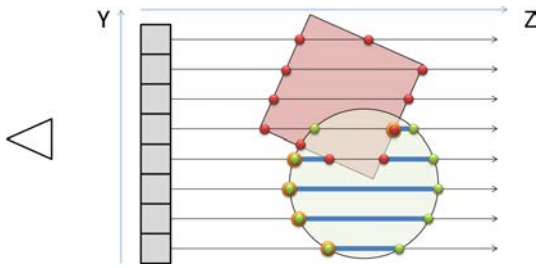


Figure 3.20 – A dexel data structure represents a solid shape as the list of its intersection points along the rays obtained by an orthographic view of the scene. This can be used to perform efficient CSG operation in a modeling software [Lefebvre 2013].

In this section we briefly present an algorithm for computing a discrete dilation in 2D. The algorithm operates on a dexel data structure (see Figure 3.20), which is used to represent the solid to dilate. Note that under this representation, an erosion is simply a dilation of the complement shape, and can be computed at the same cost as the dilation. Our algorithm has been used in a recent publication by colleagues in our team [Hornus et al. 2016]. Contrary to previous methods, it is both exact (under the discrete definition of the dilated shape) and fast, with an asymptotic complexity that does not increase with the dilation radius (in fact, the complexity of our algorithm even decreases with higher dilation radii). This is in contrast e.g. with [Martínez et al. 2015b], which can only computes dilation with kernels which are zonotopes[1], or with [Wang and Manocha 2013], which grows less efficient with larger dilation radii.

Our algorithm operates in a 2D dexel structure (which is also a row-compressed binary image). The idea is to compute the dilated shape from the Voronoi diagram of the original shape. Using every point of the original image S as seeds for the Voronoi diagram, the result $\mathcal{D}(S)$ of the dilation by a radius r is the set of (grid) points which are at distance $\leqslant r$ from any Voronoi seed.

The key insight of our is that the Voronoi diagram of a set of points in a regular grid can be computed very efficiently by doing two successive sweeps in opposite directions. Compared to Fortune's original sweepline algorithm [Fortune 1987],

[1] A zonotope is the result of the Minkowski sum of line segments in any dimension.

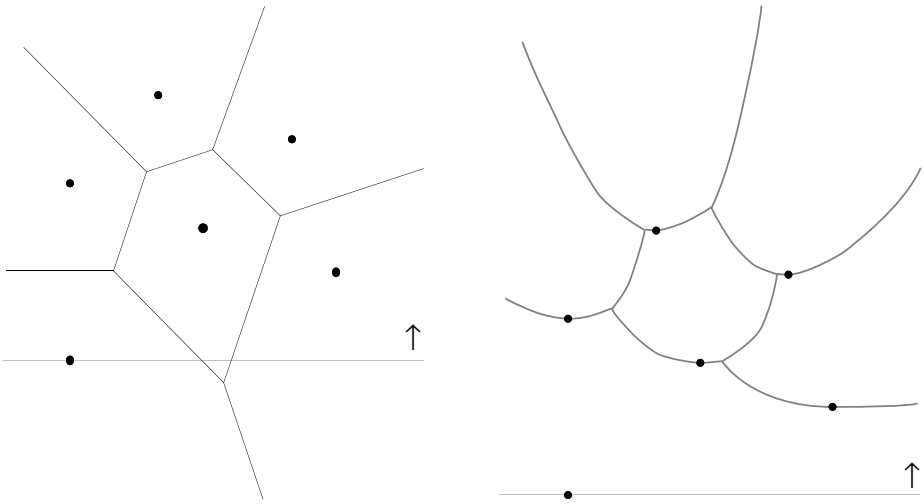


Figure 3.21 – Fortune's original algorithm [Fortune 1987] requires transforming the point coordinates to compute the correct Voronoi diagram of points in a single sweeping pass.

we can avoid the lifting illustrated Figure 3.21, which makes the algorithm simpler to implement, and possibly amenable to 3D. The idea is to use *half-space Voronoi diagrams*, where each seed is associated to a certain direction, and a seed can only "contribute" to the Voronoi diagram in the half-space corresponding to its associated direction. If all the directions are parallel, the half-space Voronoi diagram can be computed very efficiently in $O(n \log n)$ time and $O(n)$ space [Fan et al. 2011] — a complexity similar to Fortune's algorithm, but with an algorithm which is simpler to implement.

In practice, we compute two dilated shapes, one corresponding to the half-space Voronoi diagram where all seeds face the direction $(0, -1)$, and one where all the seeds face the opposite direction $(0, 1)$. As the union of two shapes is trivial to compute on a dexel data structure, this yields a very efficient algorithm for computing the dilation by a ball of radius r, the complexity of which does not increase with the actual radius value.

The process described so far only computes the Voronoi diagram of point seeds. However, the dexel data structure described Figure 3.20 only stores line segments. While it is possible to compute the Voronoi diagram of segment seeds by approximating them with point seeds (see Figure 4 in [Lu et al. 2012]), this would remove the interest of using a compressed dexel representation in the first place, as we would switch from a surfacic representation (dexels) to a volumetric one (pixels/voxels). Consequently, we aim at computing the half-space Voronoi diagram of parallel line segments (the dexels) directly in a single sweep. The process is illustrated Figure 3.22.

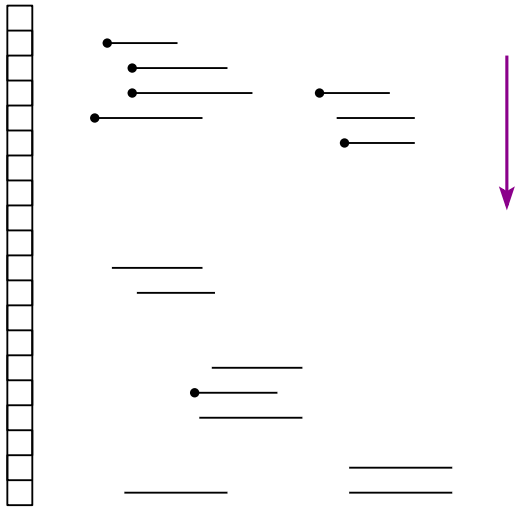


Figure 3.22 – A single sweep in one direction allows us to compute the half-space Voronoi diagram of parallel line segments (the dexel data structure).

Note that while the coordinates transformation of Fortune's algorithm would make it very difficult to compute the exact Voronoi diagram of line segments, it is straightforward to incorporate in our two sweeping passes. Indeed, we only need to compute the intersection point of the segment bisector with the sweeping line. This can be done by finding the roots of a second-order polynomial, corresponding to the curves shown in Figure 3.23.

A detailed description of our sweep-line algorithm is given in pseudo-code in Algorithms 4 and 5. In Algorithm 4, one only needs to maintain a list of *active* Voronoi cells $\mathcal{L}$, i.e. the cells that intersect the current sweep-line, and whose seeds are at a vertical distance $\geqslant r$. The event list Q indicates when a seed can safely be removed from the active list $\mathcal{L}$, i.e. when it does not affect anymore the current dilation operation. The operation DilateLine computes a dilated version of the input shape $\mathcal{S}$ intersecting the current sweep-line, given the set of active seeds $\mathcal{L}$. The operation InsertSeedSegment is described in more details in Algorithm 5. Note that in this description, we consider that segment s' is occluded by s if their projections on the horizontal line overlap. As the operation RemoveSeedSegment is very similar to InsertSeedSegment, we do not describe it in details.

In terms of data-structures, since at any point on the sweep line there can only be non-overlapping segments, the segments can be stored in a simple `std::set`, using their barycenter's x coordinate as a sorting key in the set. This provides efficient insertion and deletion of a segment in $O(\log n)$ time. It should be noted that we do not *actually* compute the Voronoi diagram of the line segments at any point in time. The `std::set` stores the seeds of the Voronoi cells intersecting the current sweep line. Moreover, if a seed segment is at a distance $>$ the requested dilation radius r, it can simply be dropped from the `std::set`. The dilated shape can then

Algorithm 4: VoronoiSweepLine

Input: A list of horizontal segments $\mathcal{S}$ (a 2D dexel structure) + dilation radius r.
Output: A list of horizontal segments corresponding to the dilated dexel $\mathcal{S}'$.

$\mathcal{L} \leftarrow \emptyset$; // Seed segments whose cell intersect the current sweep line
$\mathcal{Q} \leftarrow \{\}$; // List of events $(y,$segment id$)$ for removing seeds from $\mathcal{L}$
$\mathcal{S}' \leftarrow \emptyset$; // Dilated result
for y $\leftarrow 0$ **to** $N-1$ **do**
  foreach *event* $(y, s) \in \mathcal{Q}$ **do**
    RemoveSeedSegment$(\mathcal{L}, \mathcal{Q}, r, s)$;
  foreach *segment* $s = [(x_1, y), (x_2, y)] \in \mathcal{S}$ **do**
    InsertSeedSegment$(\mathcal{L}, \mathcal{Q}, r, s)$;
  $\mathcal{S}' \leftarrow \mathcal{S}' \cup$ DilateLine$(\mathcal{L}, y, r)$;
return $\mathcal{S}'$

Algorithm 5: InsertSeedSegment

Input:
$\mathcal{L}$ the Voronoi seeds active on the current sweep line,
$\mathcal{Q}$ the seed removal event queue,
r the dilation radius,
s the segment to insert $s = [(x_1, y), (x_2, y)]$.

Output: An updated list of active seeds $\mathcal{L}$ and removal events $\mathcal{Q}$.

RemoveOccludedSegments$(\mathcal{L}, s)$; // Remove segments entirely occluded by s
SplitPartiallyOccluded$(\mathcal{L}, s)$; // Split segments partially occluded by s
$\mathcal{L} \leftarrow \mathcal{L} \cup s$; // Insert s into the actual list $\mathcal{L}$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup (\lceil y + r \rceil, s)$; // Remove s once the sweepline has advanced beyond $y + r$
while $\exists$ *sequence* $(s_a, s_b, s) \in \mathcal{L}$ **do**
  $(x_v, y_v) \leftarrow$ VoronoiVertex(s_a, s_b, s) ; // See Figure 3.23
  if $y_v < y$ **then**
    $\mathcal{L} \leftarrow \mathcal{L} \setminus s_b$; // Remove seed s_b from the set of active seeds
  else
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup (\lceil y_v \rceil, s_b)$; // Remove s_b later on
while $\exists$ *sequence* $(s, s_a, s_b) \in \mathcal{L}$ **do**
  // Repeat operation on the right side of s

be reconstructed from the seed segments that affect the current sweep line, in linear time with respect to the surface complexity on the current sweep line.

Exact Arithmetic. An advantage of computing the Voronoi diagram of line segments whose endpoints' coordinates lie on a regular grid (the original image), is that exact arithmetic can be used to compute the Voronoi diagram and the offset. Indeed, the computation of the bisectors Figure 3.23 can be done using integer coordinates

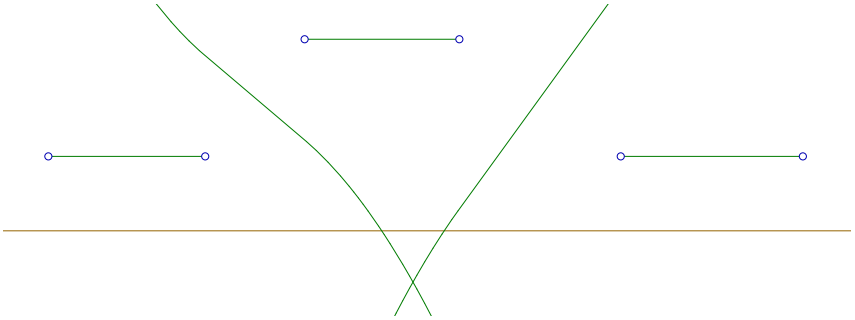
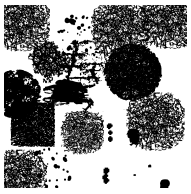


Figure 3.23 – The bisector of two line segments can be described by a piecewise second-order polynomial curve. The intersection between two bisectors are candidates for Voronoi vertices. Their $\lceil y \rceil$ coordinate determine the next event on the sweep-line that affects the current configuration of the Voronoi diagram.

(as the endpoints have integer coordinates to begin with). While the coordinates of the Voronoi vertices can have floating point values, the coordinate of the sweep line is only increased by integer steps, so the coordinates of Voronoi vertices can safely be rounded up to the next sweep line coordinate where they will change the structure of the current diagram. One caveat of using integer coordinates is that one has to be wary of possible overflows when computing the roots of the second-order polynomial describing the bisector of two line segments. This can happen on large objects which can exhibit some degenerate structures.

Future Work. While the present algorithm has been implemented and tested on 2D images, the current implementation is inherently sequential, and a parallel version — e.g. computing offsets in separate bands — has not yet been investigated. However, it should be noted that for the 2D case, the advantage of parallelism is probably limited, as the sequential version is already relatively fast on modern computers, e.g. a dilation by a ball of radius 20 pixels takes 0.515 sec on a 2048×2048 image with random complex structures, shown inset.



While the algorithm described here is works in 2D, further investigations towards a 3D extension are in order. In particular, we hope that combining sweeps in the X and Y directions with "shifted" segments would allow us to compute the exact dilation by a ball of radius r. However, the exact procedure for doing so is left as future work.

3.3 Discussion and Conclusion

In this chapter, we have presented two techniques to aid in the manufacturing process of complex shapes. The first one is a sparse support structure technique targeted at filament printers, which present the advantage of being much more stable that their tree-like counterpart, while incurring a similar cost in terms of used material and print time. Compared to dense volume-based support structures, they are usually much more economical. The second technique presented in this chapter was a fast, simple and original way to solve the problem of exact discrete offsetting of 2D and possibly 3D images, with practical applications to the computation of minimal printing volumes enclosing or enclosed in a given shape.

The use of horizontal bridges as a support structure is an original idea, and we have shown in Section 3.1 that it is competitive with other state-of-the-art methods in terms of print time, material usage, and computation cost. The inherent stability provided by the structure itself allowed us to study some additional properties, such as stability of a part during the print process. The three components of this work are rather independent subroutines: determining support points, generating the combinatorial scaffolding structure, and producing the final geometry from the description of the scaffolding. Since the publication of the original work in [Dumas et al. 2014], experience has shown that more engineering work is needed before the technique can be made available in general public slicing software. For example, we relied on a slicer-assisted definition of the support points, but other definitions could be used: local minima based on the geometry of a triangle mesh, morphological operations based on 2D slice images (see Section 2.1.3). Since our definition relies on a the vertices of the printing paths, it needs to be tightly integrated with the slicing software. This makes comparison across methods more difficult. Ideally, a modular software organization should allow the user to select how to compute the support points.

Miscellaneous remarks can be pointed out about the core scaffolding synthesis algorithm presented in Section 3.1.4. First, it is relatively independent of the printing technology, or the slicing software itself. Implementation-wise, one needs however to take care of implementing the collision detection part efficiently, as a high number of queries are being performed during the scaffolding generation. Depending on the underlying data structures — dexels or triangle mesh —, different strategies can be adopted. In Section 3.1.6, we have presented alternative formulations to solve the core synthesis problem, but the formulation being restricted on a regular grid we chose not to follow through on these formulation. Still, we hope it can inspire further research in one direction or another, if different objectives are being studied.

The last step of this work concerned the actual geometry of the structures being printed. While we empirically observed that cross-sectional pillars printed faster and more robustly than their rounded-shaped counterparts, the reality is still very complex. A thorough study would take into account influence of the type of plastic used, the print temperature, print speed, etc., which was beyond the scope of our

study. For example, a meticulous analysis on the effect of such parameters on regular objects can be found in an online blog article[1] from 3D Matter. In the context of our support structures, there is a need to evaluate more clearly the influence of the print speed — along with accelerations and decelerations —, of the *prime* and *retract* actions — when the print head starts or stops depositing matter. Clearly, printing a set of separate dots or crosses in a layer is not the same as printing a continuous path in one go, and the latter can also print faster and more reliably, because the print head does not need to stop and change speed all the time.

Finally, another aspect of the support structure generation, that we did not consider, is user guidance. Since our algorithm uses a greedy heuristic, it is easy to enrich the input with "suggestions" for potentially interesting positions to place new segments. They can be soft constraints ("*here is a cool place to add support*"), or hard constraints ("*I want a bridge* here"). The suggestions can be given by the user — through an interactive painting application —, or computed automatically — by analyzing the topological genus of the shape for instance, in order to suggest bridges that would *traverse* a loop —. This opens up multiple directions for future work, which we think could enrich the overall user experience when generating supports.

Regarding morphological operations, we have presented an alternative way to perform discrete offsetting of 2D and possibly 3D images. Notwithstanding the simplicity of computing the Voronoi diagram of a set of parallel segment with this algorithm, perhaps its most remarkable trait is the fact that its computational complexity decreases with the radius of the offset being computed. This is in contrast with existing techniques such as [Wang and Manocha 2013]. The resulting algorithm can be implemented in a few lines of code using standard STL data structures in C++. However, there are a few caveats regarding integer overflows when an exact arithmetical result is required. A parallel version of our method, as well as the 3D implementation, are left as open-problems and future work. In the meantime, I hope that this interpretation of half-space Voronoi diagrams will inspire simple implementation of other similar algorithms.

[1] http://my3dmatter.com/what-is-the-influence-of-color-printing-speed-extrusion-temperature-and-ageing-on-my-3d-prints/

Chapter 4

Shape Synthesis with Appearance Constraints

In this chapter, we study shape synthesis methods that attempt to blend *appearance* and *structural* considerations in a meaningful manner. In computer graphics, and especially in the context of texture synthesis, *appearance* is often understood as a metric upon the neighborhoods of every points in the synthesized space. By comparing synthesized neighborhoods with that of an input exemplar, the visual quality of the synthesized content can be assessed. Existing texture synthesis methods were discussed Section 2.3.1. Metrics on regular grids of pixels are not the only way appearance can be defined. For example, discrete elements with a prescribed shape can also be used, and combined — or not — with a metric on the neighborhood. This was discussed in Section 2.3.2. *Structural* considerations arise from the physical process involved when fabricating real-world models: 3D printed objects need to satisfy a number of geometrical and mechanical constraints before they can undergo the machining process. Even then, they need to withstand manipulation after printing: post-processing manipulation when cleaning support structures, day-to-day manipulation by casual users, etc. Printing constraints have been detailed in Section 2.1, while structural analysis for fabrication was presented in Section 2.2.1. Methods that seek to synthesize optimized shape from the ground up, given an objective function, have been discussed in Section 2.5.

The rest of the chapter is organized as follows. In the first section, we describe a novel by-example texture synthesis method suitable for 3D printing applications. In the context presented above, appearance is defined by pixel (or voxel) neighborhoods on the surface being synthesized, while the input exemplars are 2D images. The structural properties handled by the synthesis algorithm are: connectivity constraint, compliance (rigidity), and minimal thickness of the solid phase. Other constraints such as overhangs are treated as a post-process by printing with external support structures, such as presented in Section 3.1. This work has been presented at SIGGRAPH in 2015, and has been carried out with An LU and Sylvain LEFEBVRE at INRIA, with help from our colleagues Jun WU and Christian DICK from T.U. München. An efficient GPU implementation of the surface texture synthesizer was first realized by joint first author An LU, following an original idea of Sylvain,

before we started to study the possibility of printing the resulting patterns. I was then mostly implied in this second part of the project, where I contributed to and implemented the ideas presented in Section 4.1.4. The content presented here is reproduced from [Dumas et al. 2015], along with the usual layout changes. Please note that the texture synthesis step in Section 4.1 is also an original contribution, which already provides high-quality surface textures, comparable to state-of-the-art texturing methods. This was also the first attempt — to the best of our knowledge — at an automated optimization technique that truly takes into account both *visual* and *structural* objectives.

While the technique described in Section 4.1 is an appealing first approach, there was clearly room for improvement. Most notably, the way the structural analysis and the texture synthesizer interact is in fact quite ad-hoc: the output of one step is fed back to the other, in an iterative process, without any stronger form of constraint within each step. As a result, reinforcements suggested by the structural analysis step can become very visible, and the texture synthesizer is doing its best to make them appear seamless, but without any strong guarantee.

Consequently, I became interested in developing a different approach, which would combine texture synthesis and structural optimization in a more intertwined manner. As a result, we have developed a different technique, which is in fact closer to standard topology optimization with the SIMP approach, but with an extra term for the appearance function. The result of this research is presented in Section 4.2, and explains how both energy terms have been successfully combined in a meaningful manner, along with several extensions — such as symmetry constraints, self-weight problems, and 3D synthesis —. The content of Section 4.2 was published at SIGGRAPH ASIA in [Martínez et al. 2015a], and is reproduced here. This research was done with colleagues Jonas MARTINEZ and Sylvain LEFEBVRE at INRIA, in collaboration with Li-Yi WEI from University of Hong Kong, where I traveled two times as part of one-month long visits.

The third section of this chapter is dedicated to another form of visual criteria: the appearance is determined by discrete elements, which serve as the base building blocks for the designed shape. Discrete elements have been used successfully for texture synthesis applications in [Ma et al. 2011, 2013], and were discussed in more details in Section 2.3.2. As it turns out, recent and concurrent works in topology optimization are also considering discrete elements for structural optimization problems, as discussed more extensively in Section 2.5.3. However, the scopes and applications of these works are quite different from ours, as we target interactive design tools for 3D shape synthesis. In Section 4.3, we present our approach to the problem, which aim to be a simple, efficient, and practical solution to a problem that is very complex and computationally demanding, especially in 3D. The results presented in Section 4.3 are preliminary results — currently unpublished —, as they are the subject of research carried towards the end this thesis. However, they are included in this manuscript for the sake of completeness.

4.1 By-Example Synthesis of Structurally Sound Patterns

Remark. *Readers familiar with our original publication [Dumas et al. 2015] are invited to skip directly to the discussions and comparisons with concurrent works in Section 4.4.*

Figure 4.1 – Our by-example pattern synthesis algorithm produces structurally sound patterns along the surface of an objects, resembling the input exemplar. The reinforcements are integrated *within* the pattern, by a joint optimization of appearance and structural properties. **Top:** The Standford bunny with a variety of synthesized patterns. **Bottom:** Example patterns. These objects are printed in ABS plastic on low-cost filament printers, using a dense support. The patterns are fully connected and survived the cleaning process thanks to their reinforced structure. Yet, the reinforcements are inconspicuous as they seamlessly blend within the appearance.

Several techniques exist to automatically synthesize a 2D image resembling an input exemplar texture. Most of the approaches optimize a new image so that the color neighborhoods in the output closely match those in the input, across all scales. In this section we revisit by-example texture synthesis in the context of additive manufacturing. Our goal is to generate not only colors, but also structure along output surfaces: given an exemplar indicating "solid" and "empty" pixels, we generate a similar pattern along the output surface. The core challenge is to guarantee that the pattern is not only fully connected, but also structurally sound.

To achieve this goal we propose a novel formulation for on-surface by-example texture synthesis that directly works in a voxel shell around the surface. It enables efficient local updates to the pattern, letting our structural optimizer perform changes that improve the overall rigidity of the pattern. We use this technique in an iterative scheme that jointly optimizes for *appearance* and *structural soundness*. We consider fabricability constraints and a user-provided description of a force profile that the object has to resist.

Our results fully exploit the capabilities of additive manufacturing by letting users design intricate structures along surfaces. The structures are complex, yet they resemble input exemplars, resulting in a modeling tool accessible to casual users.

4.1.1 Introduction

Additive manufacturing empowers designers and artists with an unprecedented ability to imagine and manufacture fine, intricate patterns. The patterns may be arranged in a delicate overall structure that flows in space and suggests the surface of a larger object. The sculptures *Crania Anatomica Filigre* by artist Joshua Harker [Harker 2011], the surface autoglyphs by Segerman [2009] or the designs by company *Nervous System* [Rosenkrantz and Louis-Rosenberg 2007] are impressive and fascinating examples of this trend, also witnessed by the popularity of Voronoi carvings on the sharing platform *Thingiverse* (e.g. Chess Set - Voronoi Style `thing:172960`, Coral Candle Fixture `thing:32513`).

In this section we consider the problem of automatically generating such patterns, from an input example. Our intent is to empower casual users and designers with a tool that quickly generates a compelling pattern that prints correctly and does not break in every-day use. Performing this task by hand is very difficult: the user has to be skilled in the use of CAD systems — which are often not targeted at such models — but also needs a good understanding of the limitations of additive manufacturing as well as notions of mechanical engineering to foresee when and how the object might break.

Using our approach, the user quickly obtains a solution that enforces these constraints, and resembles the input pattern. She is then able to focus on the most important task: exploiting our technique for designing interesting and intriguing objects.

We understand this problem as an instance of *by-example texture synthesis*, a long standing problem in computer graphics [Wei et al. 2009]. Despite the wide spectrum of available methods, only a few are capable of synthesizing a pattern along a surface, and these approaches either manipulate finely tessellated models and per-vertex colors, or synthesize a volume of colors, or operate through a planar parameterization of the surface. This does not fit our purpose: we seek to produce a pattern that flows along the surface, without suffering from the distortions or discontinuities of planar mappings. Existing volume approaches are not well suited as we seek a method capable of efficiently updating the synthesis result locally, so as to reinforce and strengthen the global structure.

Most importantly, none of the existing techniques generate patterns with controlled structural properties. A simple failure case is to consider the connectivity of the synthesized pattern. Given an exemplar pattern describing a connected, single component pattern, the available surface synthesizers cannot offer any guarantee regarding the connectivity of the output. Our situation is more general as we not only seek for a connected pattern, but also for patterns that are *rigid enough* to withstand the manufacturing process and the necessary finishing steps, as well as every-day manipulation.

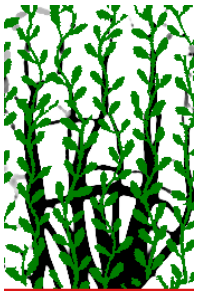
Contributions.

- A novel by-example on-surface texture synthesizer, working in a voxel shell around the surface. It synthesizes along the surface, without the need for a global parameterization, and yet does not resort on a computationally expensive volume definition of the texture synthesis problem.
- A graph abstraction of the synthesized patterns, allowing an approximate but fast computation of weak areas within the patterns, both by a stress analysis and by a geometric criterion.
- A pattern reinforcement strategy, that combines the synthesizer and a structural analysis to drive the pattern towards an object that can be fabricated.

This adds up to an algorithm for by-example synthesis of complex carved patterns along surfaces, that can be fabricated even on low-cost fused-filament printers.

Assumptions. We assume that the input surface shell can be printed correctly at the user-chosen shell thickness, *without the pattern*. Otherwise carving the pattern would only worsen its already failing structural properties. We also assume that the exemplar appearance allows for enough degrees of freedom to create connections. Our technique will produce a correct output in any case, but the appearance may be impossible to reconcile with the connectivity requirements, resulting in visible reinforcements (see Section 4.1.5).

Topology Optimization. One possible way to improve the structural soundness of a shape is via *topology optimization*. The Solid Isotropic Material with Penalization (SIMP) method [Sigmund and Maute 2013; Christiansen et al. 2015a] seems particularly well suited to our goal, since it considers a distribution of material in a grid and maximizes a rigidity objective under a prescribed material consumption ratio [Sigmund 2001].



However, there are challenges in using this approach for our purpose. In particular, the newly generated structures would significantly disturb the visual appearance of the original pattern: topology optimization tends to accumulate matter non-uniformly as shown in the Figure inset. In this figure, a pattern was first synthesized. The initial pattern (green) is then used as passive elements with fixed density, where a vertical force (gravity) is applied. The bottom nodes are fixed. The SIMP method is then used to distribute additional material (black) and reinforce the structure by minimizing its compliance. Matter tends to concentrate at the bottom. This is due to the accumulation of the forces: the regions below the loads have a much higher sensitivity to the overall compliance. This leads to the destruction of any details in these regions. Our approach avoids this

issue by reinforcing the pattern with a small number of thin bridges between nearby structures, and uses by-example synthesis to preserve the appearance everywhere.

Finally, the performance of topology optimization currently makes it impractical for the detailed geometric structures we target (on a bunny model with 734 415 voxels, 2 814 642 degrees of freedom, a single design update requires $\geqslant$ 1 hour using a multigrid solver).

4.1.2 Overview

Input. Our approach starts with a user specified target surface — given as a triangulated mesh $\mathcal{M}$, a desired shell thickness and an input exemplar pattern. The exemplar pattern is an image specifying colors as well as a binary pattern map in which pixels are tagged as either solid (1) or empty (0). This is illustrated in Figure 4.2.

The user also specifies a radius $r_{pattern}$ which captures the scale of the features in the example pattern. We assume that this scale is larger than the minimal printable feature.

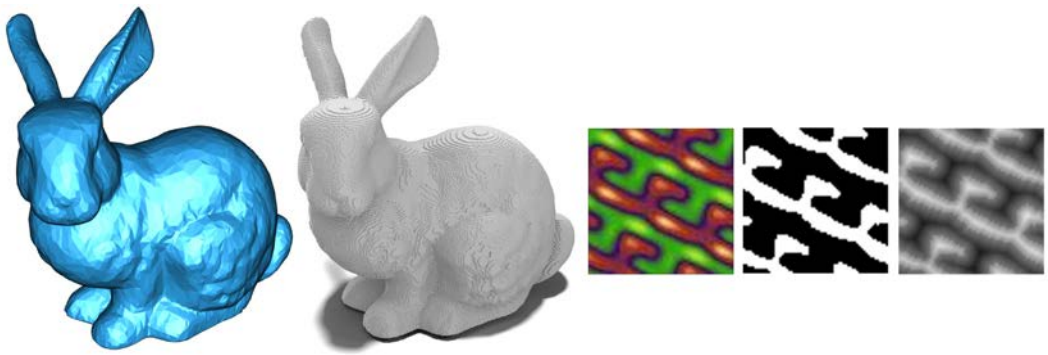


Figure 4.2 – Input: The original mesh, the voxelized surface, the colored pattern and its binary mask.

Pre-processing. As a pre-processing step we voxelize the surface shell. We consider a regular grid of voxels and select only the voxels that intersect the surface. In addition, we augment the exemplar images with a distance field computed within the binary pattern map. The feature distance is used as an additional channel when comparing the values of pixels [Lefebvre and Hoppe 2006].

Note that the voxelized surface shell has a thickness of only one voxel during the entire process. We only thicken it to the user specified thickness prior to 3D printing. Structural analysis properly takes into account the final thickness, but creating these voxels from the start would be wasteful since pattern synthesis and analysis is restricted to the surface voxels.

Algorithm. Our algorithm generates a pattern along the voxelized surface shell, that is both visually similar to the example and is structurally sound. The key novelty of our approach lies in the interplay of these two objectives. Instead of reinforcing the patterns after the fact, our scheme optimizes jointly for both objectives, incorporating reinforcements within the synthesized pattern. The difference is illustrated in Figure 4.3.

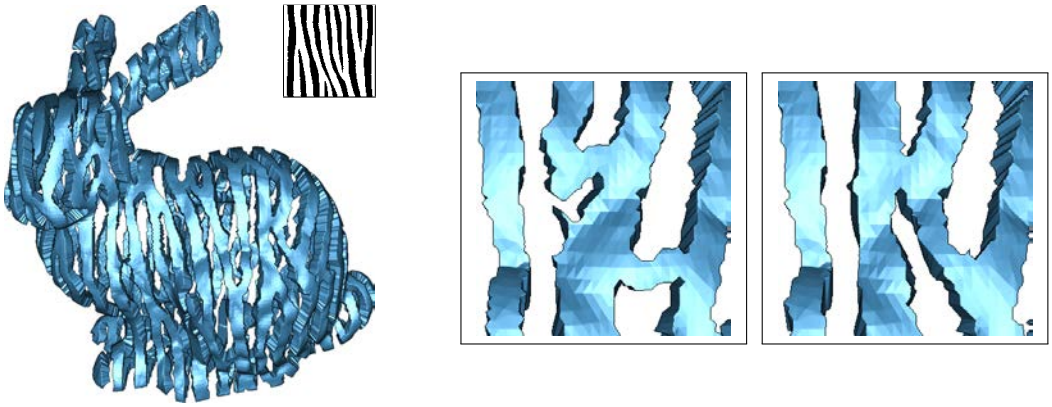


Figure 4.3 – Left: A result from our approach, and the corresponding exemplar (zebra). **Middle:** Closeup. Without resynthesis reinforcements are visible. **Right:** At the same location our approach seamlessly blends reinforcements within the pattern.

Targeting a structurally sound object means that we seek for the following properties: 1) the object can be manufactured and can survive the cleaning step after the printing process and 2) the object can withstand a user-specified *force profile*, i.e. points of attachments and external forces, most typically gravity, describing in which context the object will be used.

These two objectives are slightly different in nature. Optimizing only for a specific force profile means that the object will be sound under that particular circumstance, but it might exhibit fragilities under different conditions. We therefore incorporate a secondary objective which eliminates such fragilities, without any specific knowledge of the force profile. This is described in details in Section 4.1.4.

The overall algorithm for structurally sound, by-example pattern synthesis is given in Algorithm 6. Synthesize performs a complete synthesis pass, generating an initial pattern. It is only concerned with the appearance. StructuralOptim modifies the voxels according the structural considerations and returns the modified set of voxels and a boolean indicating whether the stopping criteria has been met ($done$). Resynthesize updates the pattern to recover its appearance, while avoiding damaging the changes made by the structural optimizer.

The synthesizer is described in Section 4.1.3 and the structural optimization in Section 4.1.4. We present our results in Section 4.1.5.

Algorithm 6: PATTERNSYNTHESIZER

Input: Surface mesh $\mathcal{M}$, input exemplar I

Output: Surface voxels tagged as solid or empty, so that the produced pattern resembles Iand is structurally sound.

```
1 V ← VOXELIZE (M);
2 V ← SYNTHESIZE(I,V);
3 while true do
4     (V, done) ← STRUCTURALOPTIM(V);
5     if done then
6         return V
7     V ← RESYNTHESIZE(I,V);
```

Notations. We denote the input exemplar image $I : (x, y) \rightarrow (r, g, b, f)$, with f the feature distance. We denote the *state* of a pixel i by $s(i) \in \{0, 1\}$, where a value of 1 means solid and a value of 0 empty. Our goal is to synthesize a mapping from any given voxel center to a location $(x, y) \in \mathbb{R}^2$ of the exemplar image I.

We denote the list of surface voxels $\mathcal{V} = [\![1, n]\!]$. Each voxel $v \in \mathcal{V}$ encloses a (small) surface patch. We assume the small patch to be planar and going through the voxel center. The voxel center position is denoted as $\mathbf{p}(v) \in \mathbb{N}^3$, its normal — averaged over the enclosed surface — is denoted as $\mathbf{n}(v) \in \mathbb{R}^3$.

4.1.3 Surface Texture Synthesis

Our synthesizer builds upon a new formulation of texture synthesis on 3D surfaces. We consider a set of planes in $\mathbb{R}^3$, each defining an orthogonal projection of the (tiled) exemplar everywhere in space. Any voxel center can thus lookup a color from any of these planes, by projection. Our synthesizer chooses in every voxel a unique source plane so that the combinations of projected colors along the surface gives the illusion of a continuous texture resembling the exemplar image. This idea is illustrated in Figure 4.4.

By controlling the set of planes, we easily guide the synthesizer towards different pattern scales or orientations. The approach is efficient as we only need to encode a choice of plane in the voxels, while the other information (coordinates, projection, colors) is implicit.

The synthesizer performs a multi-resolution synthesis into a pyramid of voxelized representations of the initial voxels $\mathcal{V}$. We denote $\mathcal{V}^r$ the resolution levels, with $\mathcal{V} = \mathcal{V}^0$ the finest level and $\mathcal{V}^L$ the coarsest level. For the sake of clarity, let us for now only consider the finest resolution level $\mathcal{V}$.

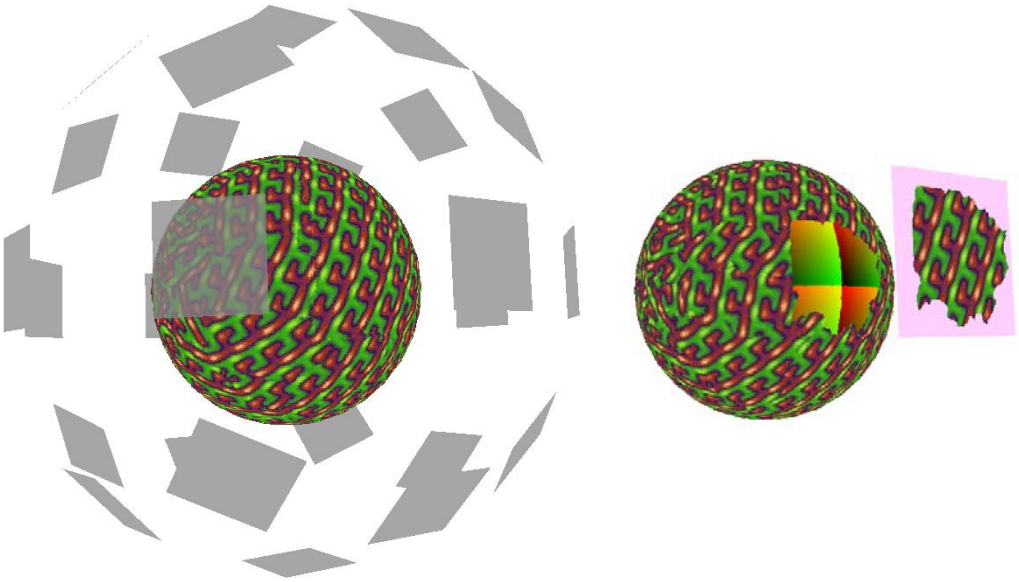


Figure 4.4 – Left: Planes surrounding the object, each defining an orthogonal projection of the texture onto the surface. **Right:** Each surface point selects a plane as the source for its color. The optimizer naturally grows patches. The image shows the source plane for one of the patches. The colors within the surface patch are revealing the (u, v) lookup coordinates into the exemplar image.

Layers Around the Surface

We consider a set of planes Ψ chosen to have normals uniformly distributed in the unit sphere. The exact location of the planes does not matter, so let us assume they surround the object as illustrated in Figure 4.4, left.

The planes define orthogonal projections of the texture onto the surface, and the voxels will receive their color from one of these planes. By optimizing these choices, the synthesized texture will appear along the surface. This is illustrated in Figure 4.4, right.

We denote by ψ a plane in Ψ of normal $\mathbf{n}(\psi)$. In addition we define a set of plane transformations Γ, each $\tau \in \Gamma$ defined by an origin point $\mathbf{o}(\tau)$ in the plane and two orthogonal vectors $\mathbf{u}(\tau), \mathbf{v}(\tau)$. Every τ maps the exemplar I to the plane in a different manner. The $\mathbf{u}(\tau), \mathbf{v}(\tau)$ vectors are not necessarily normalized so as to allow for scaling.

Each voxel $v \in \mathcal{V}$ is associated with a mapping to exemplar texture space by choosing a plane-parameterization pair $(\psi(v), \tau(v)) \in \Psi \times \Gamma$. The mapping function $M(\mathbf{x}, \psi(v), \tau(v))$ projects x on the surface of the chosen plane $\psi(v)$ via an orthogonal projection. The projected point is then mapped to image space by the planar transformation $\tau(v)$.

Our texture synthesis consists of finding a good set of choices of $(\psi(v), \tau(v))$ for voxels $v \in \mathcal{V}$ such that the resulting texture resembles the exemplar appearance.

Synthesis as an Energy Optimization

We now formalize the optimization problem to achieve texture synthesis along the surface. We start by making several observations regarding the desired properties of the result, and then give a precise formulation of the energy to optimize.

Desired Properties. Let us consider a voxel $v \in \mathcal{V}$. It corresponds to a small surface patch, approximated locally as a plane. All the surface points are mapped to the image space by the same function $M(\mathbf{x}, \psi(v), \tau(v))$ where $(\psi(v), \tau(v))$ is the parameterization choice for v, the voxel enclosing $\mathbf{x}$.

If the normal to the plane $\mathbf{n}(\psi(v))$ and the voxel normal $\mathbf{n}(v)$ align perfectly, then the resulting texture in v is a copy of one portion of the input exemplar, uniformly scaled and rotated by $\tau(v)$. This locally reproduces the exemplar appearance. However, if the normal of the plane disagrees with the voxel normal, the texture will appear distorted along the surface enclosed within the voxel. Our energy term penalizes such cases, encouraging voxels to choose projection planes agreeing with the local surface normal.

Let us now consider that there is a negligible amount of distortion, as would be the case for a planar surface. In this case, preserving the exemplar appearance boils down to achieving inconspicuous texture transitions between the voxels, e.g. similarly to image quilting [Efros and Freeman 2001]. Let us consider $5 \times 5 \times 5$ neighborhoods of voxels. We denote by $\mathcal{N}(v)$ this neighborhood for a voxel v. If the neighboring voxels perform the same choice of mapping, that is for all $w \in \mathcal{N}(v), (\psi(w), \tau(w)) = (\psi(v), \tau(v))$, then the voxels copy coherent (adjacent) texture patches, and the appearance is preserved.

If the voxel chooses different projections, color discontinuities might become visible at voxel boundaries. We measure the quality of the transition by considering the color differences between the colors that the neighborhood *expects* and the color the voxel has, and vice versa. This idea is illustrated in Figure 4.5.

Synthesis Energy. Based on this analysis, we derive the energy function that measures the quality of a choice of mappings $C : \mathcal{V} \to \Psi \times \Gamma$. The global energy is defined as the sum of two terms, one for color transitions and one for normal alignments:

$$E(\mathcal{V}, C) = \sum_{v \in \mathcal{V}} (E_{transition}(v, \mathcal{V}, C) + E_{distortion}(v, \mathcal{V}, C)) \tag{4.1}$$

(a) Projection onto the planes **(b)** 2D view of the green plane

Figure 4.5 – Mapping voxel centers to exemplar texture space. (a) Two voxels selecting different planes. (b) Projection of both voxel centers into the plane selected by v_1. The color of both projected points is used to determine whether the transition is visible. A similar operation is performed into the plane selected by v_0.

The transition error is measured as:

$$\begin{aligned} &E_{transition}(v, \mathcal{V}, C) = \\ &\sum_{w \in \mathcal{N}(v)} \left\| I(M(v, \psi(v), \tau(v))) - I(M(v, \psi(w), \tau(w))) \right\|^2 \\ &\quad + \left\| I(M(w, \psi(v), \tau(v))) - I(M(w, \psi(w), \tau(w))) \right\|^2 \end{aligned} \tag{4.2}$$

The distortion error is:

$$E_{distortion}(v, \mathcal{V}, C) = 1 - \left(\mathbf{n}(v) \cdot \mathbf{n}(\psi(v))\right) \tag{4.3}$$

Optimization Scheme

We optimize for $\widetilde{C} = \arg\min\{E(\mathcal{V}, C)\}$. To make the problem tractable we select a finite set of planes Ψ and plane transformations Γ. Therefore, each pair (ψ, τ) can be seen as a label. In the following we consider the pairs to be integers indexing planes and plane transformations.

We now describe an optimization scheme quickly selecting labels. It is inspired by the upsample-jitter-correction scheme of Lefebvre and Hoppe [2005]. Note, however, that our formulation could also be solved by other optimizers, such as alpha-expansion on the labels (ψ, τ) in a global optimization approach [Kwatra et al. 2003, 2005; Lempitsky and Ivanov 2007]. However, for stochastic textures, the greedy local improvement strategy gives good results while enabling a fast parallel update scheme [Wei et al. 2009].

We now consider the multi-resolution pyramid of voxels, together with a Gaussian stack for the input exemplar I. The algorithm is given in Algorithm 7. UPSAMPLE

propagates the choices from one level to the next, simply copying the choice of the parent in the child voxels. JITTER introduces randomness: a fraction of the voxels are forced to have random selections of labels. This percentage is exposed to the user to control how regular or random the pattern should be. OPTIMIZE performs the actual labeling, and is described in Algorithm 8.

Algorithm 7: SYNTHESIZETEXTURE

Input: Pyramid of surface voxels $\mathcal{V}^0, ...\mathcal{V}^L$
Output: Choices for each resolution level $C^0, ..., C^L$

$C^L \leftarrow \{(\psi(v), \tau(v)) = (0,0) | v \in \mathcal{V}_0\}$;
for l *from* $L-1$ *to* 0 **do**
    C^l = UPSAMPLE(C^{l+1});
    C^l = JITTER (C^l);
    C^l = OPTIMIZE(C^l);
return C^0

Algorithm 8: OPTIMIZE

Input: Surface voxels $\mathcal{V}$, set of choices C, planes Ψ and plane transformations τ.
Output: Optimized set of choices C

for $v \in \mathcal{V}$ **do**
    // Coherent candidates
    $K \leftarrow \{(\psi(w), \tau(w)) | w \in \mathcal{N}(v)\}$;
    // Random candidates
    for $i \leftarrow 1$ **to** R **do**
        (r_1, r_2) = sample a random pair in $\Psi \times \Gamma$;
        $K \leftarrow K \cup \{(r_1, r_2)\}$;
    $e_{min} \leftarrow E(v, \mathcal{V}, C)$; // Current energy
    $t_{best} \leftarrow (\psi(v), \tau(v))$; // Initialize best choice
    foreach $k \in K$ **do**
        C' = C with $\psi(v) \leftarrow \psi(k), \tau(v) \leftarrow \tau(k)$;
        $e_{min} \leftarrow \min(e_{min}, E(v, \mathcal{V}, C'))$;
        $t_{best} \leftarrow$ update best choice;
    $C(v) = t_{best}$;
return C

In every optimization step, we construct a set of candidate choices for each voxel. We first include choices from the neighbors, a process inspired by the coherent-candidate mechanism [Ashikhmin 2001; Tong et al. 2002]. These choices tend to grow coherent patches on the surface. If, however, the neighboring voxels have disagreeing normals the distortion energy will quickly increase.

To allow the optimizer to discover better transitions, both between patches and in curved areas, we add R random candidates. The space of possible pairs is however extremely large, and we therefore bias the random sampling towards the most likely candidates. We pick a random direction within a cone around the voxel normal, and select the plane in Ψ having the normal that best aligns with this direction. The parameterization τ is either chosen randomly (uniformly), or biased towards a predefined orientation to let the user control how the pattern flows along the surface.

The candidate with lowest energy becomes the choice for the current voxel. Voxels are updated in parallel, using a sub-pass update mechanism [Lefebvre and Hoppe 2005].

Pattern Synthesis

The synthesizer is used to directly produce colors into the voxelized surface shell. By considering the solid attribute in the exemplar image, we obtain a carved pattern. Of course, this pattern generally cannot be fabricated — we will discuss these aspects in Section 4.1.4.

As can be seen in Algorithm 6 there are two different calls to the synthesizer: Synthesize and Resynthesize. The first performs the initial synthesis, and the second adapts the patterns to the changes made by the structural optimizer.

Initial Synthesis. Initial synthesis is performed using our synthesis algorithm without any change, with optional user controls such as a pattern orientation.

Re-ynthesis. After structural optimization, a new set of voxels has been marked as solid to reinforce the pattern. These changes might disagree with the pattern itself. Let us consider the case of an anisotropic pattern. Since it tends to produce elongated contiguous features, the structural optimizer will very likely introduce connections between the features, orthogonal to their main orientation.

It is therefore necessary to locally recover the appearance. However, to have any hope for the process to converge we need to guarantee that the changes will not be removed entirely. We therefore perform a local update of the pattern, that is constrained in two ways. First, the voxels which have been forced as solid for structural concerns may only select candidate labels marking them as solid through the projection onto the exemplar. Second, only the surrounding voxels will be locally resynthesized. These are free to adapt to their surroundings. We employ a local constraint scheme which propagates through a few resolution levels. All parent voxels having one child voxel tagged as solid are also tagged as solid. Synthesis then resumes from the coarser resolution level to the finest, restricting the set of updated voxels to a local region located below the coarsest parent tagged as solid. The coarser resolution level is selected by going back $\log_2(r_{pattern})$ levels, where $r_{pattern}$

is expressed in voxels of finest resolution. We illustrate the process in Figure 4.6, where a red color shows the solid voxels introduced by structural optimization and the green color shows voxels which are locally re-synthesized to adapt to the constraint.

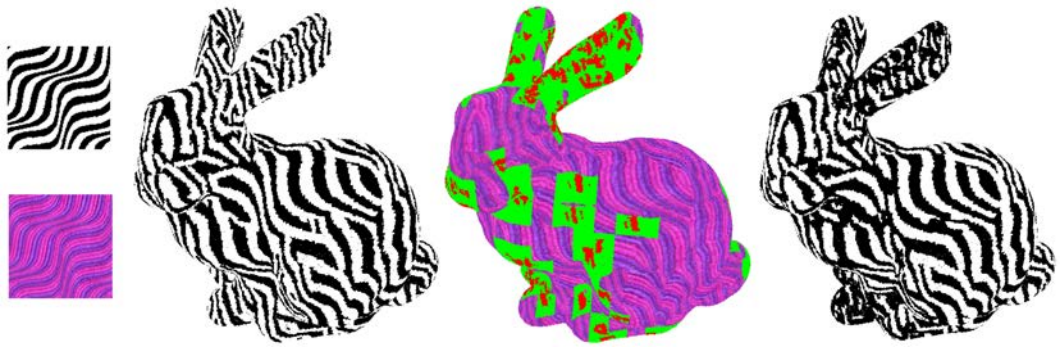


Figure 4.6 – From left to right: 1) Exemplar pattern at the top (white is solid), color version at the bottom. 2) Initial synthesis result. 3) The constraint map of a single iteration shown on the colored version of the exemplar. The red areas can only select candidates with a solid tag, while green areas are free to adapt. The rest of the surface is not allowed to change. 4) Final pattern after all iterations are completed. Note that the ears are now connected to the body, and that the neck has been strengthened.

4.1.4 Structural Optimization

After the first synthesis step the voxel states (solid/empty) obtained from the projected exemplar do not, in general, define a structurally sound pattern, and in most cases not even a single connected component. The goal of our structural optimization is to correct these issues, working jointly with the optimizer towards the final pattern.

We give in Algorithm 9 the pseudo code for the structural optimization step. GenerateSurfaceGraph and GenerateAbstractGraph are described in Section 4.1.4, and ReinforcementBridges is described in Section 4.1.4.

We assume that the full surface shell, without the carved pattern, is a strong enough object. Under this assumption, the worst case scenario of the structural optimization would be to fill all empty voxels, completely removing the synthesized pattern.

Algorithm 9: StructuralOptim

Input: Surface voxels $\mathcal{V}$ tagged as solid or empty
Output: Surface voxels $\mathcal{V}$ tagged as solid or empty with improved structural properties

```
1 G_surface ← GenerateSurfaceGraph(V);
2 G_abstract ← GenerateAbstractGraph(V, G_surface);
3 V ← ReinforcementBridges(G_abstract, V);
4 return V;
```

Surface Graph and Abstract Graph

We now need a data-structure on which to perform analysis. The number of surface voxels is generally high — in the order of 10^6 — and it is computationally prohibitive to directly manipulate this data, especially for the computation of structural properties. We therefore propose to define an abstraction of the synthesized pattern.

We define the *surface voxels graph* as the weighted undirected graph $G_{surface} = (\mathcal{V}, E)$, where $(u, v) \in E$ is an edge if and only if voxels u and v are contiguous, that is they touch by a corner: $\|\mathbf{p}(u) - \mathbf{p}(v)\|_2^2 \leqslant 3$. The edges are weighted by the euclidean distance between the voxels they connect. A closeup on a surface graph is shown in Figure 4.7 (left). Note that this graph is not impacted by the choice of solid/empty voxels and does not depend on the synthesized pattern. It is thus generated only once. For the sake of clarity we assume next that $G_{surface}$ is fully connected: the input surface mesh $\mathcal{M}$ is a single object. We otherwise treat each component separately.

The first part of our analysis process is to abstract the surface graph into a graph of lower complexity — between 500 and 1000 vertices — while still retaining the same global connectivity as the synthesized pattern. Figure 4.7 (right) shows an example of the abstract graph.

We denote by $d_G(u, v)$ the graph geodesic distance between u and v in a graph G. The *abstract surface graph* is denoted $G_{abstract} = (\mathcal{S}, \mathcal{F})$, where $\mathcal{S} \subset \mathcal{V}$. Each edge $f \in \mathcal{F}$ is tagged with a state $\rho(f) \in \{0, 1\}$ indicated whether it is considered empty or solid.

The subset of voxels $\mathcal{S}$ in $G_{abstract}$ is selected by down-sampling from the set of *solid* voxels $\mathcal{V}$ in $G_{surface}$. This is done following a Poisson disk sampling strategy, with a binary search on the radius to reach a target number of voxels (1000 in our examples) [Bowers et al. 2010]. We then connect these voxels by edges so as to reproduce the connectivity of the synthesized pattern. This is done by algorithm ConnectSamples (Algorithm 10).

ConnectSamples proceeds as follows: starting from sources in $\mathcal{S}$, it initially grows regions in G_{solid} — the graph $G_{surface}$ restricted to solid voxels. Whenever two regions are connected by the growth, the two source voxels are connected by a new edge in $G_{abstract}$. The newly added edge $f \in \mathcal{F}$ is marked as solid ($\rho(f) = 1$). In

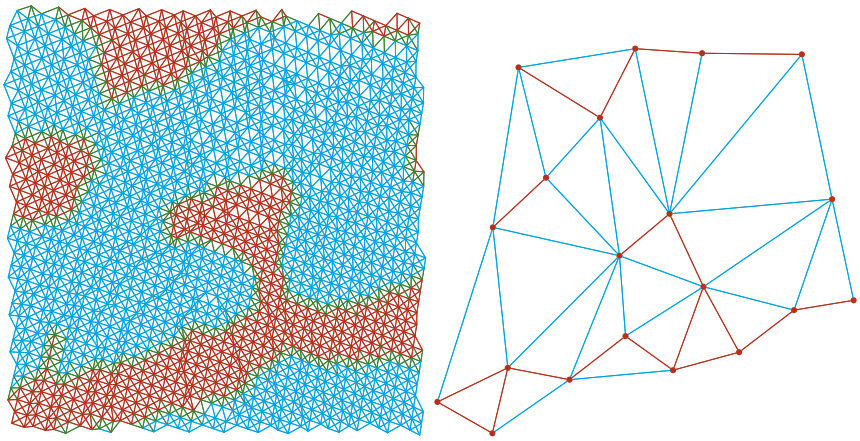


Figure 4.7 – The initial surface graph $G_{surface}$ (left) is used to produce the abstract pattern graph $G_{abstract}$ (right), having a greatly reduced number of vertices but still capturing the connectivity of the pattern. Red edges are solid and blue edges are empty.

a second pass, the same regions are grown in the whole graph $G_{surface}$. For each adjacent regions that were not previously connected a new edge is added in $\mathcal{F}$. It is marked as empty since the connecting path in $G_{surface}$ goes through empty voxels. The paths connecting source voxels through $G_{surface}$ are recorded. They are used later to produce reinforcements.

Algorithm 10: ConnectSamples

Input: The surface and solid voxels graphs and subsets $\mathcal{S} \subseteq \mathcal{V}$

Output: A graph $G_{abstract} = (\mathcal{S}, \mathcal{F})$, which is a subset of $G_{surface}$. A function $\rho : \mathcal{F} \to \{0, 1\}$ for the edge type.

1 Compute shortest path forest $\mathfrak{F}$ from $\mathcal{S}$ in G_{solid} (via Dijkstra);
2 For each pair of tree $T(u), T(v) \in \mathfrak{F}$ that have a vertex incident to the same edge $e \in E(G_{solid})$, connect their roots ($\mathcal{F} \leftarrow (u, v) \cup \mathcal{F}$), and set $\rho(u, v) = 1$;
3 Compute shortest path forest $\mathfrak{F}'$ extending $\mathfrak{F}$ in $G_{surface}$;
4 For each pair of tree $T(u), T(v) \in \mathfrak{F}'$ s.t. $(u, v) \notin \mathcal{F}$ and both trees have a vertex incident to the same edge $e \in E(G_{surface})$, connect their roots ($\mathcal{F} \leftarrow (u, v) \cup \mathcal{F}$), and set $\rho(u, v) = 0$;
5 **return** $\left(G_{abstract} = (\mathcal{S}, \mathcal{F}), \rho\right)$

Reinforcement Bridges

We now proceed to reinforcing the pattern. This step performs a number of local changes, adding new voxels. The algorithm iteratively selects edges of $G_{abstract}$ marked as empty and changes them to become solid. The corresponding voxels

in $\mathcal{V}$ are in turn switched to become solid. This information is fed back to the synthesizer.

We rely on two complementary approaches to compute the scores in the edge selection process. The first and main score is based on an analysis of stresses under a certain force profile in a simplified beam geometry capturing the pattern (Section 4.1.4). The second is a purely geometric criterion based on pattern geodesic distance in $G_{abstract}$ (Section 4.1.4).

The rationale for using two criteria is that the force profile only predicts a single scenario. Therefore, under a different set of circumstances some fragile configurations might exist. The second score acts as a worst-case criterion. Note that the user is in charge of deciding how safe he wants the print to be. Strengthening the second criterion will ultimately make the force profile have less impact. We generally allow only a few changes from the worst-case criterion, so as to obtain interesting effects from the force profile without extreme fragilities under other conditions.

A primary goal of the edge selection process is to form a single connected component in $G_{abstract}$, considering solid edges only. To this end, edges are added iteratively, choosing the next empty edge of highest score which connects two different components, using the force profile criterion. A number of additional edges are then added using the force profile criterion again, until the maximum stress of all edges is below a material-dependent threshold. We currently manually fix this threshold based on ABS/PLA plastic elastic properties. After adding edges with the force profile, we consider the second, worst-case criterion. We again mark edges as solid until no edge considered as weak remains.

The algorithm finally modifies the set of voxels. Each edge in $G_{abstract}$ corresponds to a voxel path — the one that connected the regions as described in Section 4.1.4. Adding only these voxels would not be sufficient, as it would produce a very thin bridge between two parts of the pattern. Instead, we dilate these paths and add all voxels which are within distance $r_{pattern}$ of the voxels along the path. Let us re-emphasize that adding edges in $G_{abstract}$ does not necessarily produce a straight line in $\mathcal{V}$ as it follows the shortest path between the solid regions, across empty voxels.

Score Based on Force Profile

We now exploit the abstract graph to build a simplified mechanical model of the pattern. Intuitively, each edge will become a beam whose stiffness is determined by whether it is solid or empty.

We cannot directly use truss or beam elements from the finite element method (FEM) literature, as these are 1D elements that feature a free rotation at their endpoints. This pivot joint behavior does not properly capture the printed pattern. Instead, given the abstract surface graph, we generate a 3D geometry based on hexahedral

and wedge elements, which we then simulate with the FEM to obtain a structural analysis of the pattern.

To achieve a proper geometric construction we make the assumption that the surface around each voxel is locally planar — which is correct given a smooth mesh as input. This makes it possible to represent every graph voxel by wedge elements, considering the 1-ring of neighbors projected into the local plane, as illustrated in Figure 4.8. The elements are extruded along the voxel normal, by an amount that corresponds to the desired shell thickness. Each neighbor connects to the current graph voxel by an hexahedral element incident to the face of a wedge.

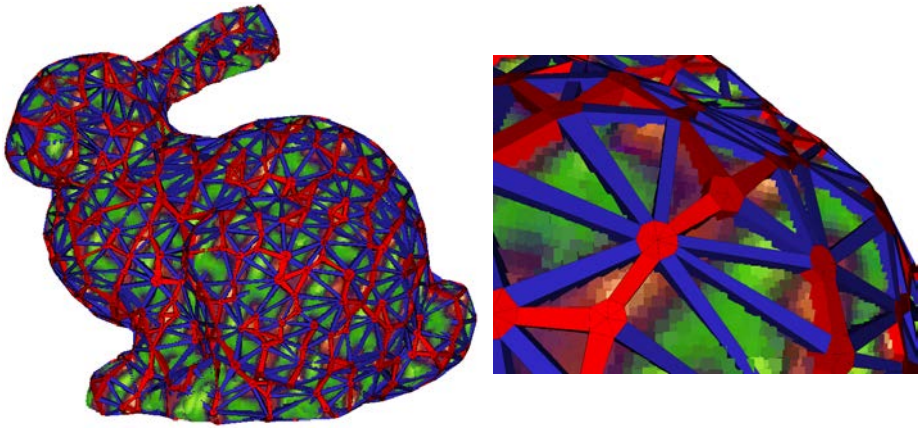


Figure 4.8 – Each vertex of the abstract graph becomes a set of wedge elements connecting to its neighbors through hexahedral beams. Solid elements are shown in red, while soft elements are shown in blue. Each beam corresponds to an edge in the graph, letting us derive a stress tensor for each edge through finite element analysis.

Note that we do not take into account flips or self-intersections that may occur if the curvature is too high. None of the models we tested created such cases.

Finite Element Analysis. The geometry constructed from the abstract graph is composed of wedges and hexahedral elements, onto which we apply the finite element method. We simulate the displacement of each node (vertex) of the beam-wedge geometry. The shape functions for interpolation within the elements are given in supplemental material. We simulate an elastic material having the properties of the printing material, typically ABS or PLA plastic.

Each element is associated with a stiffness $\rho(e)$. For wedges it is always 1, and for hexahedral elements — which correspond to graph edges — it is set to either 1 or $\rho_{\min}$ depending on whether the corresponding edge in $G_{abstract}$ is marked as, respectively, *solid* or *empty*. The stiffness matrix of the element is then derived as $\widetilde{\mathbf{K}}_e = \rho(e)\mathbf{K}_e$.

Once the global stiffness matrix is assembled we need to decide on a set of external forces and locked nodes. Let us briefly assume these are given. We can then solve the static equilibrium equation $\mathbf{Ku} = \mathbf{f}$, which we compute using the Eigen library [Guennebaud; Jacob, et al. 2010] and the sparse solver CHOLMOD [Chen et al. 2008b]. This computes a small displacement of the element nodes, which in turn allows us to estimate a stress tensor in the beam of each edge: $\sigma = \{\sigma_x, \sigma_y, \sigma_z, \tau_{xy}, \tau_{yz}, \tau_{xz}\}$, given by $\sigma_e = \mathbf{B}(\xi, \eta, \zeta)\mathbf{Eu}_e$. For more details on computing stresses with the FEM, the reader is referred to [Cook et al. 2007a]. We use the sum of the norm of the principal stresses as the score for the edges.

Solving for equilibrium is relatively expensive. We therefore insert multiple edges at once. To avoid accumulation in areas of high stress, we forbid edges close to an already inserted edge, in the manner of the dart-throwing process. The radius of cancellation is twice the length of the already inserted edge.

External Forces and Locked Nodes (Boundary Conditions). In the standard setting we only consider gravitational forces applied on the system. For each vertex $i \in G_{abstract}$, we apply a constant pressure in the direction of the gravity at position $\mathbf{p}(i)$, with an arbitrary but constant value. We fix all nodes that are within the 10% first layers in Z. This corresponds both to the fabrication process and the case where the model is standing upright and the points on the ground are fixed. Note that our approach makes no assumption about the forces and locked nodes and it is for instance possible to consider other scenarios such as pinch grips [Stava et al. 2012].

Score Based on Geometric Criterion

The geometric score is used to detect poor configurations that lead to fragilities under conditions diverging from the specified force profile, without having to resort on a mechanical simulation.

We observe that the worst fragilities consists in elongated structures that are disconnected from their surroundings. We again exploit $G_{abstract}$ to detect and suppress such configurations.

Let us consider a voxel in $G_{abstract}$ belonging to an elongated structure. This voxel is connected to a few voxels of the same structure by solid edges, and to neighboring structures by empty edges. Since at this stage the structure is fully connected, there exists a geodesic path *within the pattern* between the two voxels at each extremity of an empty edge. This is illustrated in Figure 4.9.

We consider the length of the geodesic path in G_{solid} as the score for an edge. We iteratively add all edges with a score higher than a user-defined threshold (typically 1.5 times the extent of the volume in our examples). The score of all edges is updated after each addition.

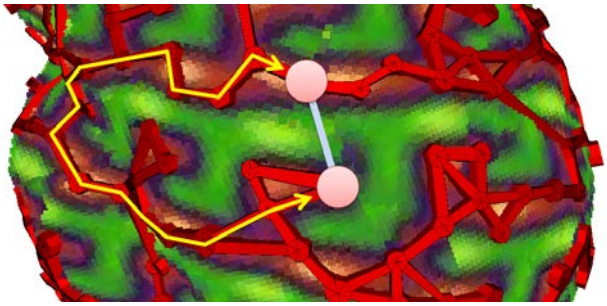


Figure 4.9 – Geodesic distance criterion. The view is a closeup of a pattern with the solid edge of the abstract graph overlaid. The outlined blue edge is an empty edge of the abstract graph being considered for reinforcement. Its extremities are connected by a shortest path through the solid edges, outlined by the yellow broken line. The score of the edge is its geodesic distance in the abstract graph $G_{abstract}$.

4.1.5 Results

Texture Synthesis

We first present results from our on-surface texture synthesizer. Figure 4.10 shows results, and Table 4.1 describes performance numbers. In all results, we use the following parameters: synthesis at 256^3 resolution (8 levels) for all but the skull result in Figure 4.14 and Figure 4.15 which uses 512^3 (9 levels). We use planes evenly distributed on the unit sphere in all 26 directions, with 16×16 translations and 64 rotations per direction. We use $5 \times 5 \times 5$ neighborhoods and 128 candidates in total, starting with coherent candidates given by defined neighbors, and filling the rest with random candidates. All shown results orient the pattern with a naive vector field around the 'up' direction. This could however be controlled by the user. We support non-tilable exemplars by penalizing pixels near boundaries [Lefebvre and Hoppe 2005].

In terms of quality it is roughly equivalent to existing on-surface synthesizers, but we have not added all the improvements from the state of the art (e.g. appearance space transform [Lefebvre and Hoppe 2006], interpolation of overlapping neighborhoods) which could further improve quality on color textures.

Our scheme is however conceptually simpler than the existing on-surface synthesis techniques (see [Wei et al. 2009], Section 4, for a survey) and fits our needs perfectly: the output is a thin shell of voxels and it enables fast local updates thus tightly integrating with our geometry modeling pipeline.

One drawback of our synthesizer is that it works under the assumption that the surface is smooth. Therefore, we can expect quality to degrade across sharp edges.

As can be seen Figure 4.11 synthesis quality remains reasonable even in challenging cases such as the skull front where there is a concentration of sharp edges.

Figure 4.10 – Texture synthesis results.

Figure 4.11 – Texture synthesis results on a object with sharp edges, high curvature areas, and complex topology.
Models: Skull (`thing:168602`), Knot (`thing:5506`).

Preparing Synthesized Patterns for 3D Printing

Enlarging Small Features. After the main loop of the program exits, the pattern we obtain is globally connected and structurally sound in regards of the abstract graph. However the abstract graph only captures the global connectivity of the pattern, and ignores its local thickness. We therefore further improve the pattern by recovering a consistent feature size in all places that are too thin.

We would like to enlarge the pattern directly *on* the surface, and not *out* of it. We propose to compute a geodesic skeleton of the solid surface pattern, and perform a dilation of the skeleton by the minimal printable feature radius, restricted to the surface shell. The final result is the union of the dilated skeleton and the original pattern.

Our approach bears similarities to the work of Liu et al. [2010], whereby the geodesic curve skeleton of the surface is computed. However, our algorithm is more relaxed because it does not preserve the exact topology of the surface. This is desired since the input pattern may contain small loops which appear through synthesis. We exclude these from the skeleton.

Our technique starts by performing a watershed transform of the graph to detect large regions and connect local maxima located within these regions [Haumont et al. 2003]. We next prune the leaves and small loops from the resulting graph. Figure 4.12 illustrates the cleaning process.

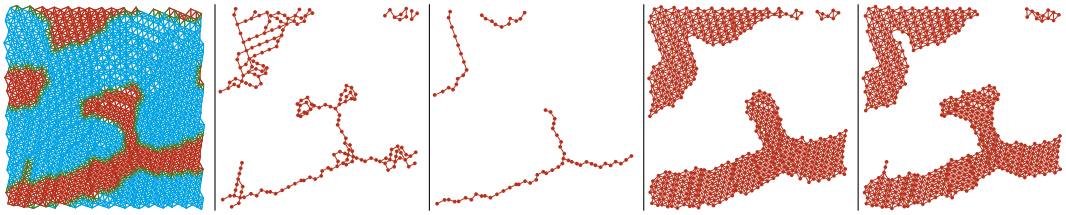


Figure 4.12 – From left to right: Initial pattern, noisy skeleton from the watershed transform, skeleton after pruning, after dilation, and after union with initial pattern. Small features have been enlarged.

Final Mesh. The contiguous solid voxels form a thin pattern along the surface shell. We thicken this shell as a post-process, by a user-specified amount. The resulting set of voxels is then meshed by extracting its orthogonal polygon which is then remeshed to obtain a smoother model.

Support Structures. We produce our models both on filament printers (*MakerBot Replicator 1*, *Ultimaker 2*) and powder based ink jet printers (*ZCorp 450*).

Our patterns are of course challenging prints on filament printers and they require support structures. We use simple supports from the slicing software, but more elaborate approaches could be used to facilitate cleanup [Dumas et al. 2014; Schmidt and Umetani 2014; Vanek et al. 2014a]. Figure 4.13 shows an object on the print bed, just after printing and the cleaned up version. Our structurally sound patterns survive the cleaning process even in such extreme cases.

Powder based printers, such as the ZCorp printers present different challenges. While support is not necessary the print is extremely fragile when removed from the powder bed — it is later strengthened by dipping it into cyanoacrylate. Figure 4.14 shows a print created on this technology.

Printing on SLS machines (laser on polyamide) would allow us to produce even thinner results without support, and would directly produce stronger parts.

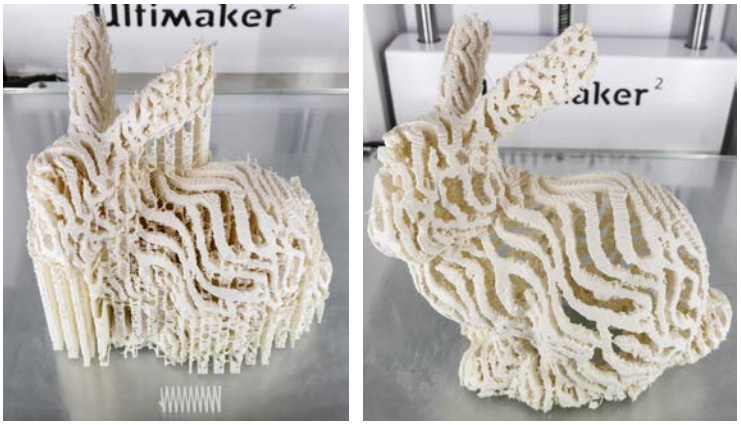


Figure 4.13 – Result before and after cleanup of the support structure.

Prints

Figure 4.1, Figure 4.14 and Figure 4.15 present 3D printed results. Note how the patterns remain easily identifiable along the surfaces. It is worth noting that isotropic patterns are generally easier to handle as their initial synthesis is often well connected, with the exception of highly curved areas, e.g. the ears of the Standford bunny model. Anisotropic patterns are more challenging as the appearance tends to conflict with the strengthening of the model. Nevertheless our technique produces natural results on different patterns.

Computation times and other statistics are summarized in Table 4.1. Table 4.2 gives performance breakouts between texture synthesis and structural analysis for the bunny model on various exemplars.

Model	Texture	Grid	# Voxels	# Iter	t_{init}	t_{total}
Bunny	Keyboard	256	67 651	4	1.11	14.6
Bunny	Bluebrown	256	84 292	10	1.11	34.8
Bunny	Greencells	256	80 917	3	1.11	11.4
Bunny	Hooks	256	55 356	11	1.11	40.0
Bunny	Waves	256	107 038	10	1.11	52.4
Bunny	Animalskin	256	89 110	24	1.30	76.2
Kitten	Hooks	256	38 626	8	0.9	24.3
Skull	Hooks	512	241 652	4	5.9	40.3

Table 4.1 – Computation time for the different models shown in the section, showing the extent of the voxelization used, number of voxels in the final geometry, number of iterations, timings for the first synthesis pass, and timings for the whole process (in s).

Texture	Synthesis	Structural Analysis	Total
Keyboard	8.6	4.6	13.2
Bluebrown	23.8	12.1	35.9
Greencells	8.3	4.9	13.2
Hooks	28.4	10.5	38.9
Waves	37.3	13.9	51.2
Animalskin	45.5	26.9	72.4

Table 4.2 – Computation times in seconds on the bunny model for the synthesis, structural analysis, and both.

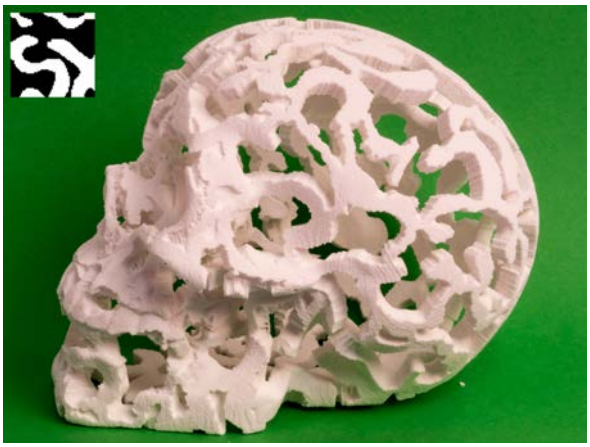


Figure 4.14 – Result printed on a ZCorp 450. On this type of machines the printed objects are very fragile and have to be extracted from the powder bed before dipping into cyanoacrylate. Our thin patterns nevertheless print successfully.

Experimental Verification

We experimentally verify our approach by simulating our output pattern with a full-scale high resolution finite element analysis. Each voxel of the thickened pattern becomes a cubic (hexaheadral) element. We use the same boundary conditions as for our optimizer: the object is fixed to the ground and subject to gravity. The material parameters are those of ABS plastic.

It is worth noting that the FEA solver requires two to five minutes to solve for the equilibrium equation. Our system does several iterations of structural optimization, each solving several times the equilibrium equation (see Section 4.1.4 and Table 4.1). It would thus be impractical to rely on the full simulation of the pattern directly.

The full FEA solution lets us validate our approach of using the abstract graph and simplified beam geometry. We first simulate the fully connected structure, before any additional edges are added. This is the first structure that can be simulated

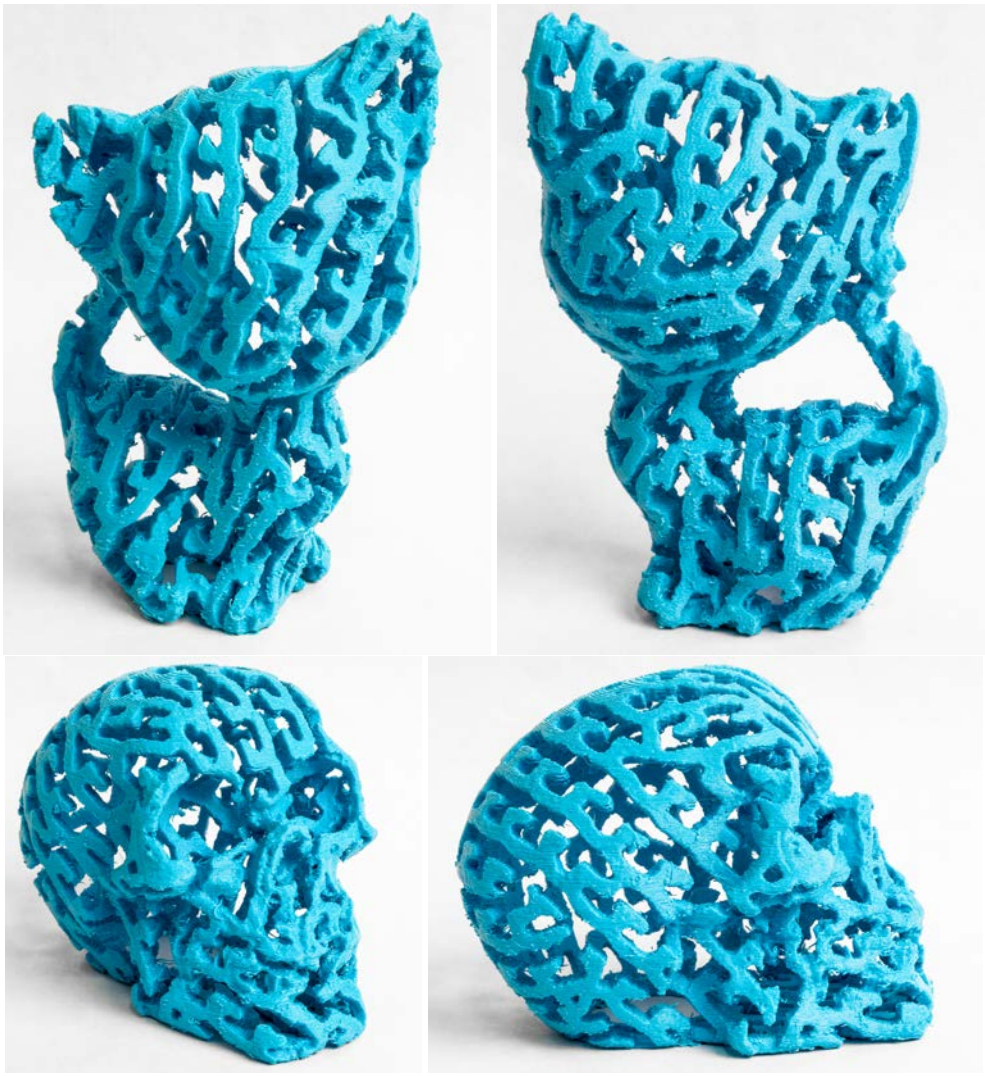


Figure 4.15 – Kitten and skull model carved with the hooks pattern.
Models: Kitten (`thing:12694`), Skull (`thing:168602`).

as disconnected structures lead to an under-determined system in FEA simulation. Second, we simulate the structure reinforced with only the force profile criterion (Section 4.1.4). Finally, we simulate the structure reinforced with our complete approach, that is force profile and geodesic criteria (Section 4.1.4). We repeat the FE analysis but this time change the force profile — the weight is applied in a direction orthogonal to the gravity used for the synthesis. This puts the pattern in an unexpected situation, for which it was not designed.

Table 4.3 and Figure 4.16 summarizes the results. As can be seen, in the expected scenario (top rows) the force profile criterion strongly reduces stresses in the structure. The geodesic criterion only marginally improves this result. In the unexpected

scenario however, the geodesic criterion further improves the result as its reinforcements compensate for the now incorrect force profile the pattern was optimized for. Note that the number across both rows cannot be compared directly as the boundary conditions differ. Only numbers within a same row are comparable.

We show in Figure 4.17 the effect of using different force profiles on the bridges added by the structural optimizer.

Scheme	Connected	Forces only	Forces + Geodesic
Stress	Expected force profile: weight along $\vec{\mathbf{g}} = (0, 0, -1)g$		
Average	16 388.5	5680.4	5418.16
Median	3961.5	3686.61	3444.95
99th%	153 947	31 720.7	**30 486.6**
Stress	Unexpected force profile: weight along $\vec{\mathbf{g}} = (1, 0, 0)g$		
Average	87 039.1	23 509.8	20 111.5
Median	20 770.5	10 374.6	9762
99th%	693 773.5	185 775	**145 854**

Table 4.3 – Stresses for a model 100mm long, using a grid of extent 256. The score shown is $|\sigma_1| + |\sigma_2| + |\sigma_3|$, in Pascal. Note that with the unexpected force profile, the 99th percentile is further decreased by the last optimisation scheme.

Limitations

Our technique works well as long as the scale of the pattern is relatively small compared to the object. This is in general the intended use, but combined to the print size limitation it would sometimes be desirable to generate coarser pattern. High curvature areas also pose multiple difficulties: texture synthesis becomes more challenging, the local planarity assumption for building the finite elements might be violated (Section 4.1.4), and thickening to obtain the final mesh may lead to fold-overs. Thus, final quality can degrade on surfaces with highly curved features (e.g. front of the skull in Figure 4.15).

Not all patterns can be used to define meaningful reinforcements, e.g. if the features of the input pattern are too thin to be printed at the selected scale. Additionally, when the input pattern is completely disconnected the appearance severely conflicts with the structural objectives. This still produces correct models, but the reinforcements are less inconspicuous as they do not blend within the pattern. Such a failure case is shown in Figure 4.18.

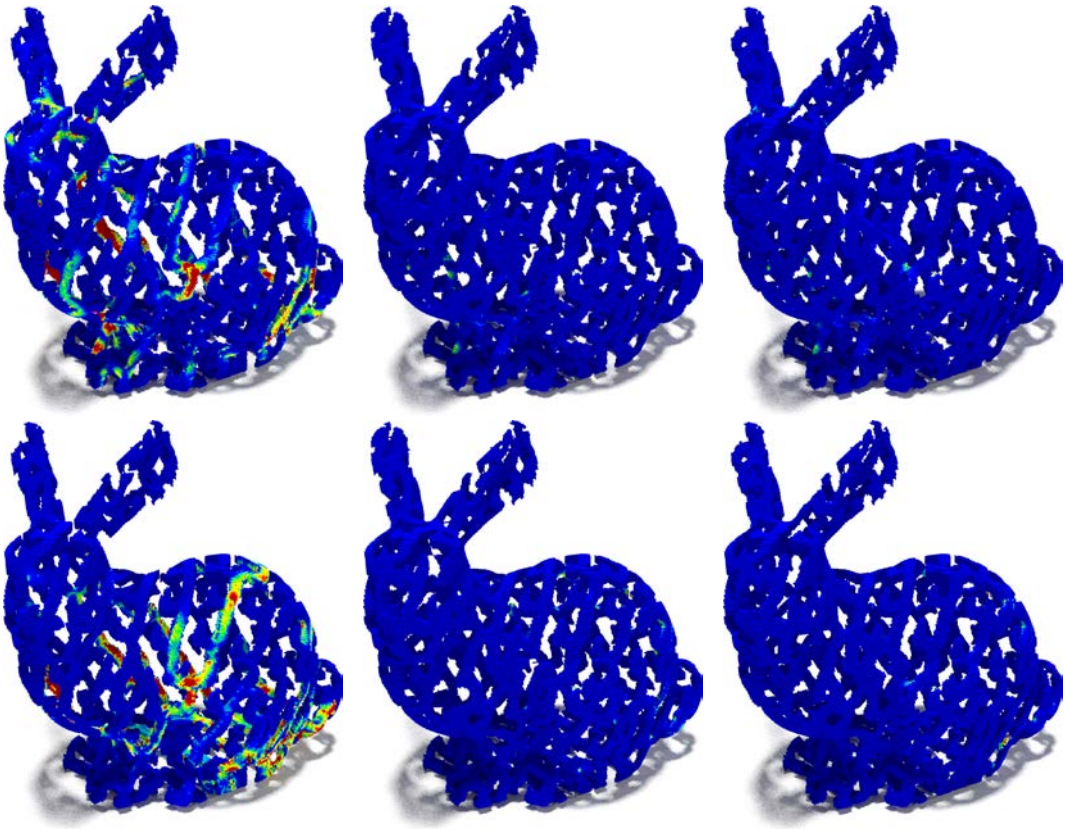


Figure 4.16 – Top: Color coded stresses for the first row of Table 4.3. The scale is the same for all three images. **Bottom:** Color coded stresses for the second row of Table 4.3. The scale is the same for all three images.

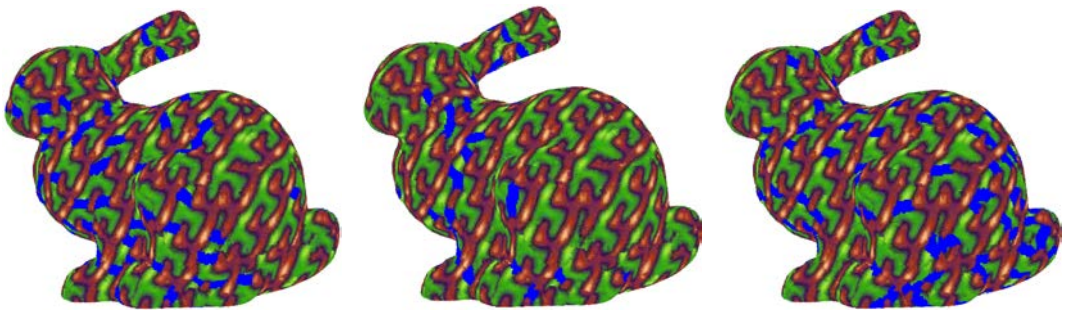


Figure 4.17 – From left to right: Different force profiles, with gravity to the left, to the bottom, and to the right of the bunny. Note how the reinforcements suggested by the structural optimizer adapt.

Figure 4.18 – This pattern is fully disconnected and therefore the appearance cannot be reconciled with the connectivity requirements of a sound structure. The produced object nevertheless prints correctly, and does capture some of the original features.

4.1.6 Supplemental

In this supplemental material, we start by giving more details regarding the computation of element stiffness matrices for our stress analysis, and explicit out approach for the enlargement of small features. We also show more results using different patterns in Figure 4.20, with source exemplar and patterns shown in Figure 4.19. All these patterns are fully optimized and could be fabricated. Finally, Figure 4.22 shows more results for texture synthesis only (using inputs shown in Figure 4.21) on a more challenging set of meshes (having sharp edges, highly curved areas, and complex topology respectively).

Shape Functions and Stiffness Matrices

Let (ξ, η, ζ) be the position of a point in an element, in the *intrinsic* coordinate system of the element. For hexahedra the corners are at positions $(\xi_i, \eta_i, \zeta_i) \in \{-1, 1\}^3$. For wedges the corners have positions $(0, 0, \zeta_i)$, $(1, 0, \zeta_i)$ and $(0, 1, \zeta_i)$, with $\zeta_i \in \{-1, 1\}$.

For hexahedral elements:

$$N_i(\xi, \eta, \zeta) = \frac{1}{8}(1 + \xi_i\xi)(1 + \eta_i\eta)(1 + \zeta_i\zeta) \tag{4.4}$$

For wedge elements:

$$\begin{aligned} N_1 &= \frac{1}{2}(1-\xi-\eta)(1-\zeta) \\ N_2 &= \frac{1}{2}\xi(1-\zeta) \\ N_3 &= \frac{1}{2}\eta(1-\zeta) \\ N_4 &= \frac{1}{2}(1-\xi-\eta)(1+\zeta) \\ N_5 &= \frac{1}{2}\xi(1+\zeta) \\ N_6 &= \frac{1}{2}\eta(1+\zeta) \end{aligned} \tag{4.5}$$

The local stiffness matrix of an element is integrated numerically over the geometry by 6-point Gauss quadrature for wedges, and 8-point Gauss quadrature for hexahedra. For more details on the assembly of the stiffness matrix please refer to a book on the finite element method [Cook et al. 2007a].

The local stiffness matrix is given by:

$$\mathbf{K}_e = \int_{V_e} \mathbf{B}^\intercal \mathbf{E} \mathbf{B} \, |J| \, \mathrm{d}\xi \, \mathrm{d}\eta \, \mathrm{d}\zeta$$

Where $\mathbf{E}$ is the 6×6 constitutive matrix that relates stress and strain. As we use isotropic linear material for our elements, the matrix $\mathbf{E}$ can be expressed as:

$$\mathbf{E} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} \tag{4.6}$$

$\mathbf{B}$ is the strain-displacement matrix (6×24 for hexahedra, 6×18 for wedges), and $|J|$ is the absolute value of the determinant of the Jacobian matrix that maps intrinsic coordinates to world coordinates.

Enlarging Small Features

Initial Skeleton Extraction. Let us denote the boundary of the solid voxels by $\partial\mathcal{V}_{solid}$. We compute the geodesic distance from each solid voxel to any boundary voxel, i.e. $d_{\partial\mathcal{V}}(i) = \min_j d_{G_{solid}}(i, j)$. This is done efficiently in a single pass of the Dijkstra's algorithm starting from $\partial\mathcal{V}$. We then compute the *watershed transform* of

the graph G_{solid} [Haumont et al. 2003]. It is defined via a flooding process, which intuitively proceeds as follows: voxels which are local (non-strict) maxima for $d_{\partial\mathcal{V}}$ serve as sources $\mathcal{S}$, with respective height given by $-d_{\partial\mathcal{V}}$. Water is then flooded through the edges, which serve as a pipes, allowing the water to reach higher heights — the difference of altitude being given by the edge weights. The watershed transform can be easily computed by Dijkstra's algorithm from the source voxels $\mathcal{S}$. The height of a voxel is the minimum distance from a voxel in $\mathcal{S}$.

When two trees spanning from different sources are adjacent by an edge in $E(G_{solid})$, they are connected together, forming voxel paths connecting sources in $\mathcal{S}$. The set of all paths forms the the watershed transform of the graph that we denote G_{wsh}.

Pruning. The resulting graph looks close to the desired skeleton but suffers from spurious structures appearing due to the discrete settings. We propose a combinatorial cleaning approach which performs well on our patterns.

To understand our pruning process, let us start with two remarks. First, we might consider recursively pruning leaves (vertices of degree 1). However, the process has to stop before pruning the main part of the skeleton as well. Second, it is not alway possible to find a leaf, and we often observe unwanted small spurious loops attached to a longer structure. We have to find where to cut open these loops and prune the resulting leaves recursively.

For the pruning process we defined for each vertex $i \in V(G_{wsh})$ the pruned distance $d_{pruned}(i)$ as

$$d_{pruned}(i) = \max\{d_{\partial\mathcal{V}}(i), \max_{j \in pruned(i)} d_{pruned}(j) + w(i,j)\}$$

where $pruned(i)$ is the set of pruned neighbors of i. Intuitively, this gives the length of the pruned branch that emanated from i.

The pruning process is summarized in Algorithm 11. The score innerCount(x) is defined as the minimum number of edges between x and another vertex of degree $\neq 2$. The rationale behind this score is, in case of a cycle, to cut open first the vertex that is further away inside that cycle. This extends the intuitive notion for the case of vertices of degree 2 to a more general tie-breaking rule. α is a constant controlling the length of loops which are considered small. In practice we select $\alpha = \pi$.

Algorithm 11: PRUNEGRAPH

$Q \leftarrow V(G_{wsh})$ sorted by lexicographic order as defined by the tuples $(d_{\partial\mathcal{V}}, \text{innerCount}, d_{pruned})$;
$T \leftarrow \{v \in V(G_{wsh}),\ deg(v) = 1\}$; `// Current leaves`
$\mathcal{P} \leftarrow \emptyset$; `// Set of pruned vertices`
while $Q \neq \emptyset$ **do**
  while $T \neq \emptyset$ **do**
    $x \leftarrow \text{pop}(T)$;
    if $d_{\partial\mathcal{V}}(x) \leqslant r_{pattern}$ *and* $d_{pruned}(x) \leqslant \alpha \cdot r_{pattern}$ **then**
      PRUNEVERTEX(x);
  if $Q \neq \emptyset$ *and* $\texttt{top}(Q) \notin \mathcal{P}$ **then**
    $x \leftarrow \text{pop}(Q)$;
    if $d_{\partial\mathcal{V}}(x) \leqslant r_{pattern}$ *and* $d_{pruned}(x) \leqslant \alpha \cdot r_{pattern}$ **then**
      if TESTPRUNELOOP(x) **then**
        PRUNEVERTEX(x);

Algorithm 12: PRUNEVERTEX(x)

$\mathcal{P} \leftarrow P \cup x$;
$G \leftarrow G - x$; `// Remove vertex from current graph`
forall *vertices* y *adjacent to* x *before pruning* **do**
  Update $d_{pruned}(y)$ and innerCount(y);
  if $\deg(y) = 1$ **then**
    $T \leftarrow T \cup y$;
  else
    $Q \leftarrow T \cup y$;

Algorithm 13: TESTPRUNELOOP(x)

Input: `maxLoopCount`
forall *vertices* y *adjacent to* x **do**
  $R(y) \leftarrow y$; `// Root in Union-Find data structure`
for *outgoing vertices* y *by increasing distance from* x **do**
  `// This is again a Dijkstra routine`
  if *a loop of length* $\leqslant$ `maxLoopCount` *connects two vertices* $u, v \in \mathcal{N}(x)$ **then**
    Merge($R(u), R(v)$);
return *Number of remaining trees in* R *is* $\leqslant 1$

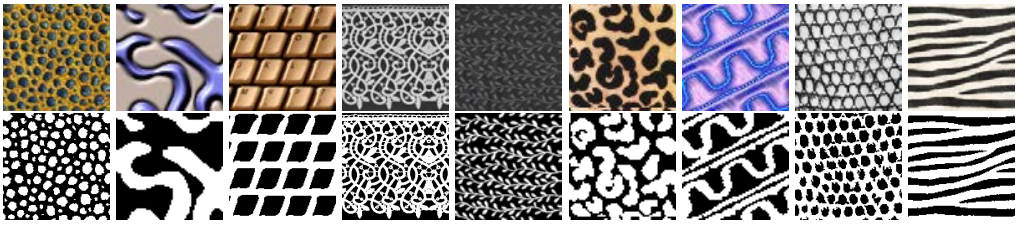


Figure 4.19 – All exemplars.

Figure 4.20 – Additional pattern synthesis results.

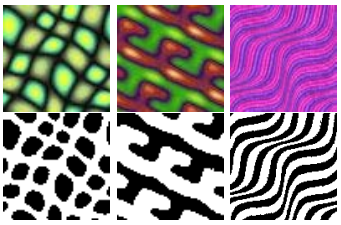


Figure 4.21 – All exemplars.

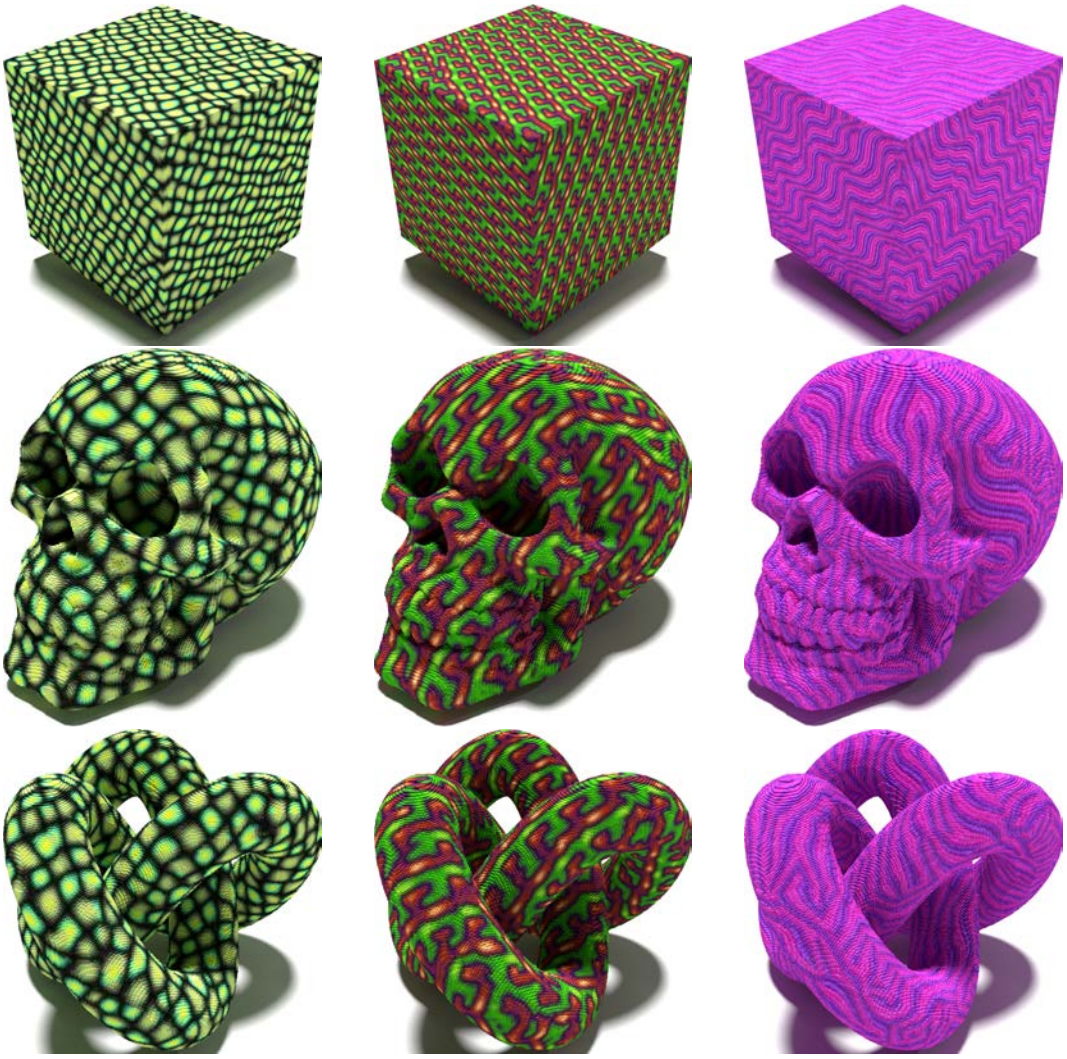


Figure 4.22 – Additional texture synthesis results on other models.

4.1.7 Conclusion

We have introduced the first method to synthesize patterns along a curved surface from an example, while ensuring that the pattern is printable and withstands a user specified force profile. We believe our work opens interesting questions regarding the joint optimization of shape structural soundness and shape appearance. By-example approaches offer unique advantages in this context: our method is easily accessible to casual users, and simple to use through the specification of a 3D model and an image of a pattern.

There are several directions of future work. A first direction is to transpose what we applied on surfaces into a volume synthesis context. The challenges are different: in 3D patterns have more opportunities to connect without violating the appearance specified in a 2D exemplar. However, the computational cost grows dramatically when dealing with volumes. A second direction is to further explore the possible controls. We currently under-exploit the ability of our on-surface synthesizer to orient and scale the synthesized textures. It is also possible to locally change the texture, introducing progressive variations along a surface (e.g. [Zhang et al. 2003]). Both controls can be combined with structural optimization, e.g. orienting the pattern locally to maximally absorb stress.

4.2 Structure and Appearance Optimization for Controllable Shape Design

Remark. *Readers familiar with our original publication [Martínez et al. 2015a] are invited to skip directly to the discussions and comparisons with concurrent works in Section 4.4.*

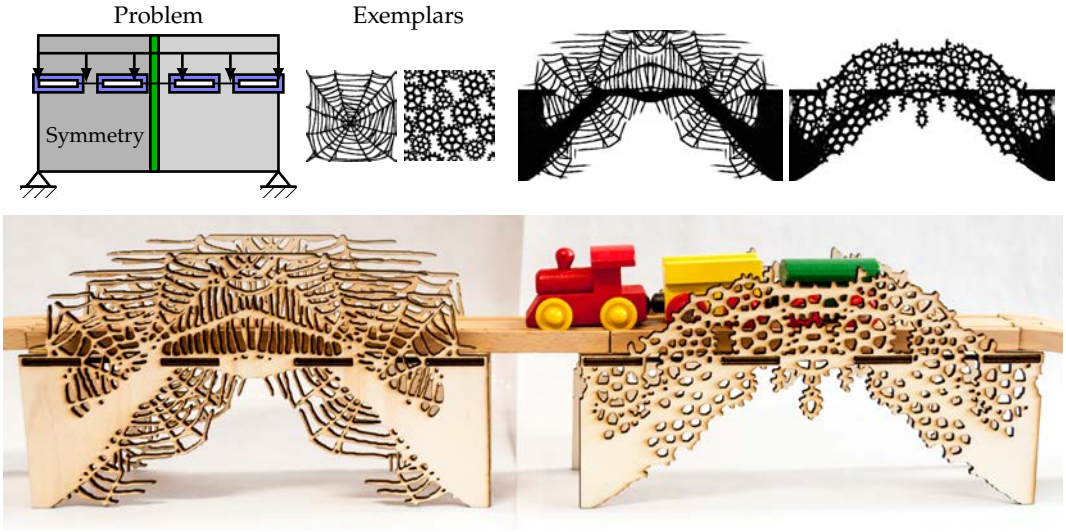


Figure 4.23 – Our technique automatically generates rigid shapes answering a specific loading scenario and resembling an input exemplar pattern, while using a user-specified quantity of material. **Top left:** Loading scenario; in this case the synthesized shape is anchored to the ground by its bottom left/right corners, while supporting a road through four attachments. Each attachment contains an empty region (white) surrounded by a solid boundary (blue), serving as a socket to plug in the road plank. **Top middle:** Two exemplars defining the desired appearance. **Top right:** Two synthesized bridges answering the loading scenario but each using a different exemplar. **Photograph:** Fabricated objects using the synthesized shapes.

The field of topology optimization seeks to optimize shapes under structural objectives, such as achieving the most rigid shape using a given quantity of material. Besides optimal shape design, these methods are increasingly popular as design tools, since they automatically produce structures having desirable physical properties, a task hard to perform by hand even for skilled designers. However, there is no simple way to control the appearance of the generated objects.

In this section, we propose to optimize shapes for *both* their structural properties *and* their appearance, the latter being controlled by a user-provided pattern example. These two objectives are challenging to combine, as optimal structural properties fully define the shape, leaving no degrees of freedom for appearance. We propose a new formulation where appearance is optimized as an objective while structural properties serve as *constraints*. This produces shapes with sufficient rigidity while

allowing enough freedom for the appearance of the final structure to resemble the input exemplar.

Our approach generates rigid shapes using a specified quantity of material while observing optional constraints such as voids, fills, attachment points, and external forces. The appearance is defined by examples, making our technique accessible to casual users. We demonstrate its use in the context of fabrication using a laser cutter to manufacture real objects from optimized shapes.

4.2.1 Introduction

Recent years have witnessed a significant spread of rapid manufacturing technologies, such as 3D printing and laser cutting. In principle, these techniques empower casual users with the ability to create tangible objects from their virtual counterparts. In practice, it remains extremely difficult to design objects which are aesthetically pleasing and at the same time structurally sound for real world constraints, such as being rigid enough to perform their intended function.

An important effort towards simplifying the creation of complex yet functional objects emerged from the field of topology optimization [Bendsøe 1989; Sigmund 2009a; Brackett et al. 2011]. In this field, the primary consideration is to design lightweight structures that are as rigid as possible. That is, optimizing for the most rigid shape using a prescribed amount of material. This is a key engineering problem as material use and weight are directly related to cost and efficiency. These techniques are a perfect match to additive manufacturing technologies as they typically produce complex geometries impossible to manufacture otherwise.

However, these approaches only consider rigidity as an optimization objective, and the appearance of the final object cannot be controlled besides explicit constraints such as avoiding regions of space or enforcing symmetries [Kosaka and Swan 1999]. In this work we propose to jointly optimize for the rigidity *and* the appearance of the structure, as defined by a user-specified exemplar pattern. This is different from after-the-facts reinforcement of the final result [Stava et al. 2012], and from synthesizing uniform, manufacturable patterns [Dumas et al. 2015] (Figure 4.24): the optimized shape is obtained as the result of a *single optimization problem* integrating both appearance and rigidity, and operates under a *constrained material budget*. It is easy to use via by-example specification and simple constraints such as solid and void regions to enforce, as well as external forces. These constraints are general and allow us to optimize for appearance, mechanical strength, and material cost.

A natural intuition is to combine topology optimization [Sigmund 2009a] and by-example texture synthesis [Wei et al. 2009] to satisfy both structural and appearance objectives. However, these are challenging to combine, preventing the algorithm to properly converge. This is confirmed experimentally (see Figure 4.27) as finding good compromises with a simple combination of these two objectives requires tedious parameter tuning, if possible at all.

Contributions. Our contributions are:

- A new formulation in which appearance is optimized as an objective while rigidity is understood as a *constraint*.
- Controls that are powerful yet easy to understand: single parameter for the appearance-rigidity tradeoff (α), volume usage bounds (v_{min}, v_{max}), and appearance specified by example.
- First order derivatives for the appearance objective, enabling gradient descent under non-linear constraints.

Scope. In this section, we target the synthesis of flat (and possibly curved) shapes that are manufactured with laser cutting. Our formulation directly translates to volumes (see Section 4.2.4), but results in large problems which are too slow to solve via our current implementation. Scalability to higher dimensions is an important direction of future work.

Foreword. In Section 4.1 [Dumas et al. 2015], we proposed a technique synthesizing a uniform stochastic pattern *covering* a surface, while ensuring that it is printable. However, the application is different: our approach in this section generates a shape under a *prescribed material budget* and will generally not fill a domain. Instead, it seeks for the optimal compromise between appearance and rigidity while distributing material in space. Figure 4.24 highlights the differences: while Section 4.1 targets the equivalent of uniform texturing, this section targets controllable shape design.

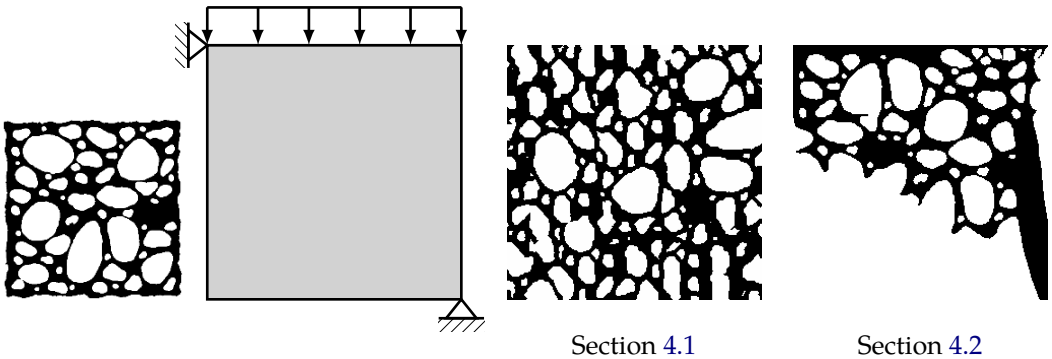


Figure 4.24 – Comparison with Section 4.1 [Dumas et al. 2015]. **From left to right:** input exemplar; loading scenario, attachment points and optimization domain (gray square); result of Sections 4.1 and 4.2. Regardless of the loading scenario the method in Section 4.1 always seeks to produce a structure that *fills* a given region, while we generate a rigid shape using the user-specified quantity of material and resembling the input exemplar.

4.2.2 Problem Formulation

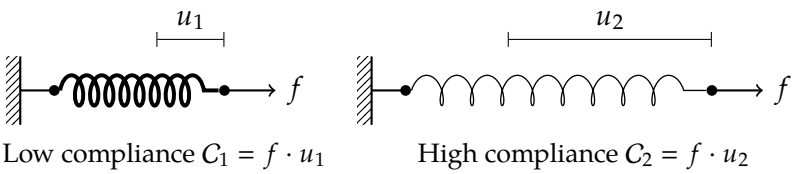


Figure 4.25 – Two 1D linear elements (springs) with a single degree of freedom. A same force f generates a displacement of respectively u_1 and u_2. The compliances reflect that the stronger spring (leftmost) is more rigid: $C_1 < C_2$. The same concept translates to elements of higher dimensions.

Our method combines two fundamental ingredients. The first is the notion of appearance as defined by neighborhood similarities between a synthesized structure and an example pattern. The second comes from mechanical engineering and is the notion of *compliance*. Figure 4.26 provides an overview of our method.

The basic problem in which compliance appears is the prediction of the mechanical behavior of a structure when it is subjected to precise boundary conditions — that is, a set of attachment points and external loads applied to the structure. In this work we consider small deformations for which the behavior of the structure can be characterized by linear elasticity. We consider isotropic materials described by their Young's modulus and Poisson's ratio. The compliance is the work exerted by the forces on the structure, i.e. the sum of the dot product of forces and displacements, as illustrated in Figure 4.25. A low compliance implies that forces produce only small displacements, which characterizes a high rigidity.

We model the shape in an n-dimensional grid of square elements denoted by $\boldsymbol{x}$, each having 2^n corner *nodes* shared with their neighbors. Each element e in $\boldsymbol{x}$ receives a *density* x_e which through optimization has to converge towards void (= 0) or solid (= 1), thus defining an interior and exterior. In practice, intermediate values remain after optimization, and we apply a thresholding after convergence.

We formulate our goal as a multi-objective optimization problem that minimizes both an appearance energy $\mathcal{A}_I(\boldsymbol{x})$ and the structural compliance $C(\boldsymbol{x})$:

$$\overline{x} = \arg\min_{\boldsymbol{x}} \left(\mathcal{A}_I(\boldsymbol{x}), C(\boldsymbol{x})\right), \tag{4.7}$$

where I is the input exemplar — a black and white pattern defining void (pixel = 0) and solid (pixel = 1) regions; $\overline{x}$ is the outcome — densities defining a shape in the grid — computed through our optimization procedure. The user has to specify at least one attachment point for the problem to be well-posed. She can optionally impose additional conditions, such as regions of void or fill, symmetry, and external forces; see results in Figure 4.23 and Section 4.2.5.

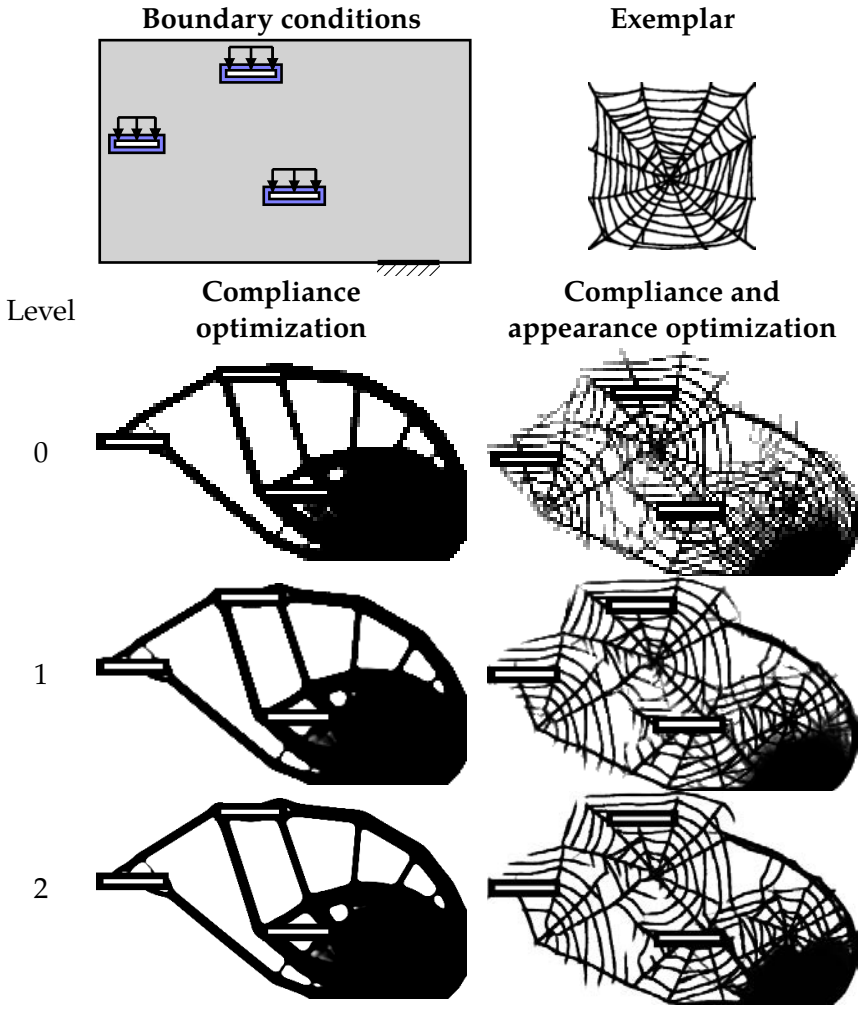


Figure 4.26 – Overview of our multi-resolution optimization approach. Given an input exemplar (upper right) and output boundary conditions (upper left), our method optimizes the corresponding output in a multi-resolution fashion (lower rows). The boundary conditions can include support, solid, and void. In this example, the volume is constrained to 35% of the overall output domain, and the relaxation factor of the compliance with respect to the optimal is set to $\alpha = 1.2$.

Previous works exist to optimize each of these energies in isolation. Therefore, a straightforward approach would optimize for a linear blend of both energies, i.e.

$$\mathcal{A}_I(\boldsymbol{x}) + \lambda C(\boldsymbol{x}), \tag{4.8}$$

with $\lambda > 0$ allowing to explore the tradeoff. Unfortunately, such a simple scheme does not produce reliable results: the values of λ that can produce a reasonable output differ widely between exemplars, boundary conditions, and domain size, when they exist at all. Figure 4.27 illustrates this issue and compares to our formulation.

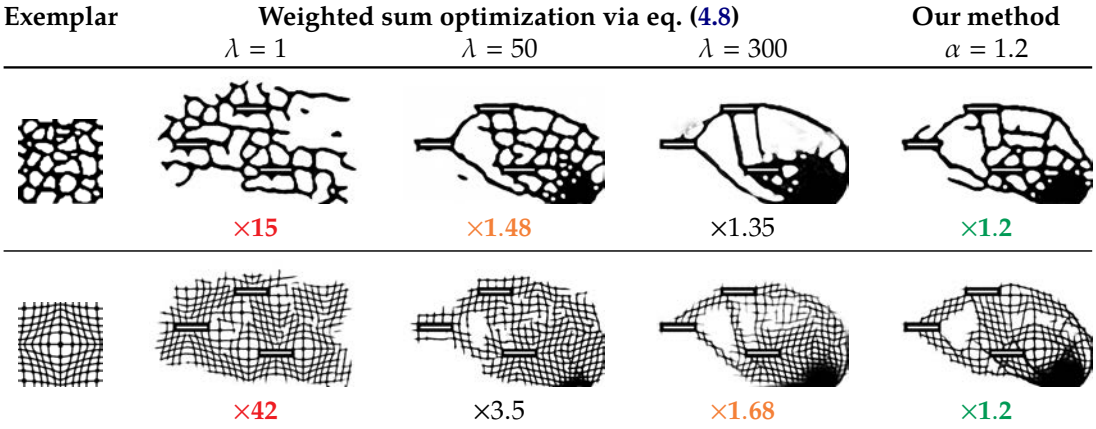


Figure 4.27 – Comparison of a straightforward weighted sum approach and our formulation. All results use the same parameters and a volume constrained to 30% of the entire domain. We give below each result the ratio between its compliance and the compliance of the shape optimized without appearance objective (C_{opt}) ; e.g. ×1.2 implies the result is within 20% of the computed optimum. λ weights the importance of rigidity versus appearance. On the left hand side, a low λ gives results with good appearance but mediocre compliance. On the right hand side, a large λ produces more rigid results but a degraded appearance. We show in orange the values of λ producing reasonable compromises, and in green and red the best and worst compliance ratios, respectively. Note that these differ significantly between both exemplars. Our method (rightmost column) does not need any specific setting besides the threshold from the computed optimum (20%). The boundary conditions are the same as in Figure 4.26.

We therefore propose to modify the formulation of the problem. We note that the goal is not necessarily to obtain the most rigid structure, but rather a structure with *sufficient rigidity*, i.e. which does not yield under the given loads. Thus, our insight is that rigidity should be considered as a constraint, which can be relaxed to allow more freedom for the appearance objective. Thus, our goal is now to minimize $\mathcal{A}_I(\boldsymbol{x})$ such that the structural compliance is below a threshold C_{max} and the volume is bounded:

$$\begin{aligned} \arg\min_{x}: \quad & \mathcal{A}_I(\boldsymbol{x}) \\ \text{subject to}: \quad & C(\boldsymbol{x}) \leqslant C_{max} \\ & v_{min} \leqslant \sum_e x_e \leqslant v_{max} \\ & \forall e \; 0 \leqslant x_e \leqslant 1 \end{aligned} \tag{4.9}$$

The volume bounding constraint is important. The weight of the structure is often negligible compared to external forces, in which case the most rigid shape would tend towards a full block of material. v_{max} prevents this naive solution to exist. On the contrary, when only considering the weight of the structure — i.e. solving a *self-weight problem* — the naive solution is an empty shape. v_{min} prevents it. The volume constraint is also a natural control for the user. Combined to appearance it allows changing the overall size of the structure (see Figure 4.28). For the sake of clarity we express volume constraints as a percentage of the design domain. We often use only external forces or only self-weight in which case we respectively set $v_{min} = 0\%$ or $v_{max} = 100\%$. In such cases we report only the non-trivial bound.

A meaningful value of the C_{max} constraint is crucial to ensure a feasible solution. We describe how the threshold is computed in Section 4.2.2, and describe the appearance objective in Section 4.2.2. We discuss our solver and numerical scheme in Section 4.2.3 and extensions in Section 4.2.4.

Compliance Constraint

We determine the compliance threshold C_{max} by first computing a solution C_{opt} to the problem considering compliance alone. We adopt the well-established Solid Isotropic Material Penalization (SIMP) topology optimization method [Bendsøe and Kikuchi 1988]. Implementation details of this method can be found in [Sigmund 2001; Andreassen et al. 2011].

Topology Optimization for Determining C_{opt}

The SIMP method seeks to minimize compliance by assigning densities to each element in $\boldsymbol{x}$ given a constrained total material budget $v_{min} \leqslant \sum_e x_e \leqslant v_{max}$. The elastic structure is simulated with the Finite Element Method (FEM).

Each square element e receives a continuous scalar density $0 \leqslant x_e \leqslant 1$, and its Young's modulus is defined as

$$E_e = E_{min} + (x_e)^p (E_0 - E_{min}) \tag{4.10}$$

where E_0 is the material Young's modulus, $E_{min} > 0$ is a small value to prevent numerical instabilities, and $p = 3$ is the standard SIMP penalization factor which penalizes intermediate values in the solution.

The compliance of the output is measured by summing the compliance of each individual element. Let us denote:

$\mathbf{u}$	Global displacement vector
$\mathbf{u}_e$	Local displacement vector per element
$\mathbf{f}$	Global force vector
$\mathbf{K}$	Global stiffness matrix
$\mathbf{K}_0$	Element stiffness matrix with unit Young's modulus
t	Element thickness

Compliance minimization is formulated as

$$\begin{aligned} C_{opt} = \quad & \min_{x} : C(x) \\ \text{subject to :} \quad & v_{min} \leqslant \sum_e x_e \leqslant v_{max} \\ & \forall e, 0 \leqslant x_e \leqslant 1 \end{aligned} \tag{4.11}$$

where the compliance term is given by the equation

$$\begin{aligned} C(x) = \quad & \mathbf{u}^\intercal \mathbf{f} = t \sum_e E_e \mathbf{u}_e^\intercal \mathbf{K}_0 \mathbf{u}_e \\ \text{subject to :} \quad & \mathbf{K}\mathbf{u} = \mathbf{f} \end{aligned} \tag{4.12}$$

We use the Method of Moving Asymptotes (MMA) [Svanberg 1987] to minimize $C(x)$ and obtain a solution C_{opt}.

Note that when the boundary conditions include large external forces, the weight of the structure tends to have a negligible influence, and can be ignored. We discuss in Section 4.2.4 how self-weight can be taken into account.

Setting the Threshold C_{max}. We first perform standard topology optimization to compute a lower bound on the achievable minimal compliance C_{opt}. We then set $C_{max} = \alpha C_{opt}$, where $\alpha \geqslant 1$. As we increase the value of α, we obtain results having a higher compliance, but that are more similar to the exemplar. This is a simple and predictable parameter controlling how much freedom is allowed to appearance, as can be seen in Figure 4.28.

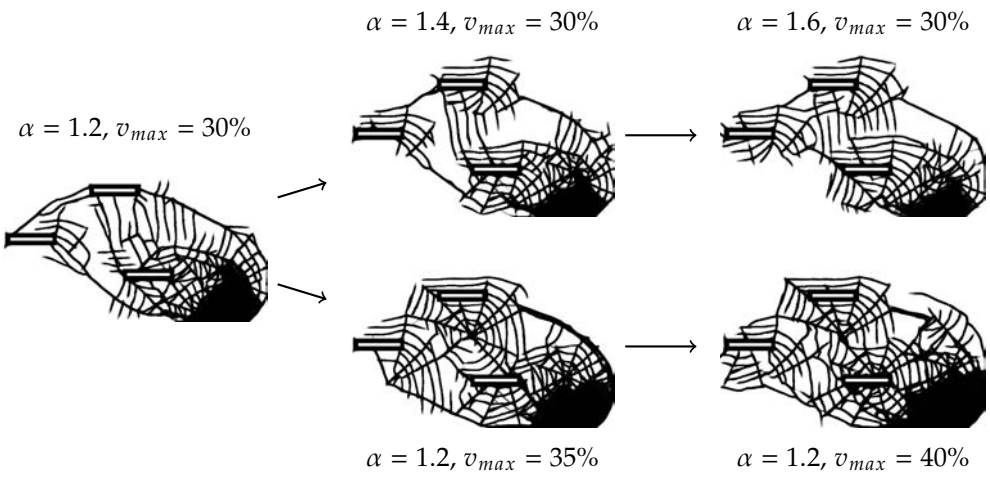


Figure 4.28 – Progressive relaxation of the maximum compliance (top row) and maximum volume (bottom row) constraints.

Appearance Objective

Our method can be used with any appearance energy that has first order derivatives available, as required by the solver (Section 4.2.3). Our formulation builds upon the work of Kwatra et al. [2005] but adapts it to facilitate the computation of first order derivatives with respect to $\boldsymbol{x}$.

In the following, when using the notation x_e, e is meant as a coordinate within the grid of elements. We denote by $N = 2\delta + 1$ the texture synthesis neighborhood size typically 15×15, i.e. $\delta = 7$). We denote by $m(e)$ the *coordinate* of the matching pixel in I whose neighborhood $\mathbf{z}_e$ is the most similar to the current neighborhood of x_e in the result. The matching process is described in Section 4.2.3. We denote by $z_f^e = I[m(f) + (e - f)]$ the value at coordinate e from the best matching neighborhood of f, where $I[z]$ accesses the pixel value (0 or 1) at coordinate z in I. The appearance energy contributed by an element $x_e \in x$ is:

$$\mathcal{A}_I(x_e) = \sum_{f\,,\,\|f-e\|_\infty \leqslant \delta} \left| x_e - z_f^e \right|^r \tag{4.13}$$

The total appearance energy is $\mathcal{A}_I(\boldsymbol{x}) = \sum_e \mathcal{A}_I(x_e)$. We use $r = 1.2$, set experimentally as in [Kwatra et al. 2005].

4.2.3 Solver

We are now ready to solve for the global optimization problem, that is to minimize appearance under the compliance constraint (Section 4.2.2). This is a challenging

optimization as all terms are non-linear and are subject to inequality constraints. In addition, the appearance objective contains combinatorial terms: the best matching neighborhoods.

We thus optimize for the density $\boldsymbol{x}$ in an iterative block coordinate descent scheme alternating between finding the best matching neighborhoods coordinates $m(e)$ and the densities x_e for each element e.

Initialization. We set $x_e = \min(\frac{v_{max}}{|\boldsymbol{x}|}, 1)$ for all elements and select random coordinates for best matches $m(e)$. One can choose a different initial guess, but a better convergence is typically observed when starting from a uniform gray; see Figure 2 in [Sigmund and Maute 2013].

Neighborhood Matching. Given the densities, the coordinates of the most similar neighborhoods are computed with PatchMatch [Barnes et al. 2009] and the random walk of [Busto et al. 2010]. For a neighborhood at coordinate e in $\boldsymbol{x}$ and a neighborhood at coordinate e' in I the similarity used for matching is defined as:

$$d(e, e') = \sum_{f\,,\,\|f-e\|_\infty \leqslant \delta} \left| x_f - I[e' + (f - e)]) \right|^r + \lambda_{occ} \frac{O[e']}{N^2} \cdot \frac{|I|}{|\boldsymbol{x}|N^2} \tag{4.14}$$

The first term compares pairwise densities throughout neighborhoods and corresponds to the appearance energy (eq. (4.13)). The second term is used to increase the spatial uniformity of the appearance energy [Kopf et al. 2007; Kaspar et al. 2015]. O is an occurrence map storing how many times each exemplar pixel is used in the different closest neighborhoods. It is computed from the previous iteration. $\lambda_{occ} \geqslant 0$ controls the amount of enforced spatial uniformity. $\frac{|I|}{|\boldsymbol{x}|N^2}$ is a normalizing factor making λ_{occ} independent of synthesis resolution. In our implementation $\lambda_{occ} = 20$.

Optimizing Densities. Optimizing the appearance objective (eq. (4.13)) given the best matching neighborhoods is more challenging. We are facing a non-linear, non-convex optimization problem for both objective and constraints. In addition, evaluating the compliance constraint is computationally expensive as it requires solving for the FEM equation. For these reasons we rely on the Globally Convergent Method of Moving Asymptotes (GCMMA) [Svanberg 1995, 2002]. GCMMA converges in fewer iterations than augmented Lagrangian methods, reducing the number of required FEM solutions. It iteratively solves subproblems that are convex approximations of the original problem, and rely on the gradient of both objective and constraints functions to do so. GCMMA therefore requires the first order derivatives of the objective and constraint functions.

Derivatives. The derivative of the volume constraint is 1. The compliance derivative is [Bendsøe and Sigmund 2004]:

$$\frac{\partial C}{\partial x_e} = 2\mathbf{u}_e^T \frac{\partial \mathbf{f}}{\partial x_e} - \mathbf{u}_e^T \frac{\partial \mathbf{K}}{\partial x_e}\mathbf{u}_e \tag{4.15}$$

The term $\frac{\partial \mathbf{f}}{\partial x_e}$ is null when the weight of the structure is neglected. For self-weight problems (Section 4.2.4) this term will however influence the result.

The derivatives are typically filtered to prevent numerical issues, and to control the optimization quality. We consider the extensively used smoothing operator described by [Sigmund 1997] to filter the derivative of the function C:

$$\widehat{\frac{\partial C}{\partial x_e}} = \frac{1}{\max(\epsilon, x_e)\sum_i w_{e,i}} \sum_i w_{e,i} x_i \frac{\partial C}{\partial x_i} \tag{4.16}$$

where $w_{e,i} = \max\big(0, \gamma - \mathrm{dist}(e,i)\big)$ is a weight factor (convolution), controlled by a parameter γ (filtering radius), and ϵ (10^{-3}) is a small coefficient avoiding division by zero.

At this step, all z_f^e are assumed to be constants. Therefore, the derivative of $\mathcal{A}_I(\boldsymbol{x})$ can be expressed as

$$\frac{\partial \mathcal{A}_I}{\partial x_e} = \sum_{f\,,\,\|f-e\|_\infty \leqslant \delta} r \frac{\left|x_e - z_f^e\right|^r}{\left(x_e - z_f^e\right)} \tag{4.17}$$

The derivative is not defined when $x_e = z_f^e$. Thus, using a small $\varepsilon > 0$, we regularize $\mathcal{A}_I(\boldsymbol{x})$ as follows:

$$\widetilde{\mathcal{A}}_I(x_e) = \sum_{f\,,\,\|f-e\|_\infty \leqslant \delta} \left(\left(x_e - z_f^e\right)^2 + \varepsilon\right)^{r/2} \tag{4.18}$$

and it follows that the derivative is

$$\frac{\partial \widetilde{\mathcal{A}}_I}{\partial x_e} = \sum_{f\,,\,\|f-e\|_\infty \leqslant \delta} r\left(x_e - z_f^e\right)\left(\left(x_e - z_f^e\right)^2 + \varepsilon\right)^{r/2-1} \tag{4.19}$$

Multi-Resolution. For improved performance and quality we optimize through a multi-resolution scheme. The process starts from downsampled versions of the grid x and exemplar I. The resolution is iteratively doubled, using the previous

result to initialize the next finer resolution by bilinear up-sampling. The process is illustrated in Figure 4.26.

Our algorithm optimizes three resolution levels. The compliance relaxation parameter α remains constant throughout the process. The exemplar is downscaled to match the resolution of each level. The strain-displacement and constitutive material matrices are also changed according the resolution.

We use the same multi-resolution scheme to compute the compliance solution (Section 4.2.2) and obtain C_{max} for each resolution level. In practice, we observe that the value of C_{max} is remarkably stable across resolutions.

Convergence. The optimization process ends when $\frac{\|\nabla \mathcal{A}\|}{\|\mathcal{A}\|}$ is below a small threshold (we use 0.001), or when a maximum number of iterations is reached (in our implementation we use $40, 20, 10$ on the three successive resolution levels).

4.2.4 Extensions

Optional constraints can be added to our basic formulation. We describe two important ones. The first is to consider the weight of the structure itself during optimization (Section 4.2.4). This is useful when there is no external force besides gravity applied to the structure. The second is to consider symmetry constraints, which are useful for aesthetics purposes but also to reduce computation time when the solution is known to have symmetries (Section 4.2.4). We also describe how to optimize for 3D outputs even though our method is dimension agnostic (Section 4.2.4).

Self-Weight

We optionally take into account the weight of the structure and the forces it generates under gravity. Note that on self-weight problems — i.e. no external forces — the complete void is a trivial optimal. We therefore impose $v_{min} > 0$ in such cases.

Forces due to the structure weight are modeled by a vertical force acting on each grid node q as follows:

$$q_x = 0, \qquad q_y = -g \sum_{e\,,\, q \in \mathbf{q}_e} x_e \frac{m_e}{|\mathbf{q}_e|} \tag{4.20}$$

where $\mathbf{q}_e$ is the set of nodes belonging to element e (as defined in Section 4.2.2), m_e is the element mass, and g is the absolute value of the gravitational acceleration. Let us emphasize that this force depends on the current densities of the elements x_e.

When using the SIMP formulation on a problem taking into account the weight of the structure, the displacements might become unbounded for low density regions, resulting in numerical issues [Bruyneel and Duysinx 2005]. We therefore use a modified formulation of material stiffness as suggested by [Pedersen 2000] to overcome this problem:

$$E_e = \begin{cases} E_{min} + (x_e)^p (E_0 - E_{min}) & \mu < x_e \leqslant 1 \\ E_{min} + x_e \mu^{p-1} (E_0 - E_{min}) & 0 < x_e \leqslant \mu \end{cases} \tag{4.21}$$

In our experiments we set $\mu = 0.25$. This switches to a linear stiffness model in regions of low densities. The derivatives are updated accordingly. The non-differentiable point where the switch between models occurs does not have a detrimental impact in practice [Bruyneel and Duysinx 2005].

We observe that on self-weight problems the volume bounds v_{min}, v_{max} have to allow for some freedom to achieve convergence. Indeed, the optimized shape is a subtle tradeoff: adding matter makes some regions more rigid but also adds stress to others through gravity. Figure 4.29 illustrates results obtained on self-weight problems.

Symmetry

Symmetry plays an important role in aesthetics for shape design. We adopt the symmetry reduction approach of [Kosaka and Swan 1999] for topology optimization. The design domain $\boldsymbol{x}$ is partitioned into $S > 1$ subdomains $\boldsymbol{x}^i$. We define a mapping between $\boldsymbol{x}^i$ and an imaginary domain $\boldsymbol{x}^*$, and only optimize for $\boldsymbol{x}^*$. The derivatives are given by:

$$\frac{\partial C}{\partial x_e^*} = \frac{1}{S} \sum_i \frac{\partial C}{\partial x_e^i} \tag{4.22}$$

That is, we compute the derivatives for the design domain $\boldsymbol{x}$, and optimize $\boldsymbol{x}^*$ according the averaged derivatives given by the mapping. Note that even though all x_e^i variables map to the same x_e^*, their individual gradients on the right-hand side may differ due to asymmetric loading scenarios. Figure 4.30 illustrates a problem solved with symmetry reduction.

Optimizing 3D Structures

Our formulation is amenable to 3D, adding a third dimension and using a grid of cubic (hexahedral) elements. While this would provide a full volume synthesis, such an approach is computationally expensive and requires a 3D exemplar as input (schemes using several 2D exemplars to define a volume could be adapted [Wei et al. 2009]).

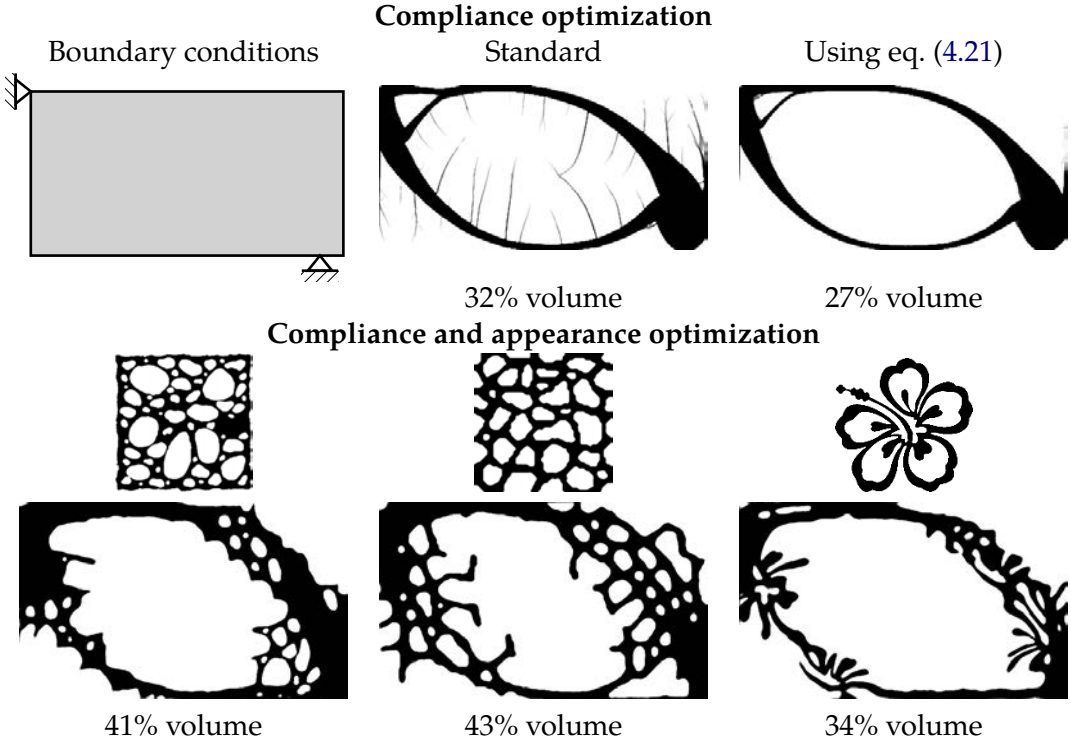


Figure 4.29 – Optimizing self-weight problems (no external forces). **Top:** A different material stiffness model is required to avoid degeneracies in low density regions. **Bottom:** All results use the same parameters $\alpha = 2.2$, $v_{min} = 20\%$, $v_{max} = 100\%$. Self-weight problems are more challenging to optimize (see text), and therefore the bounds are relaxed to let the optimizer converge. Note how different volumes are obtained depending on the exemplar.

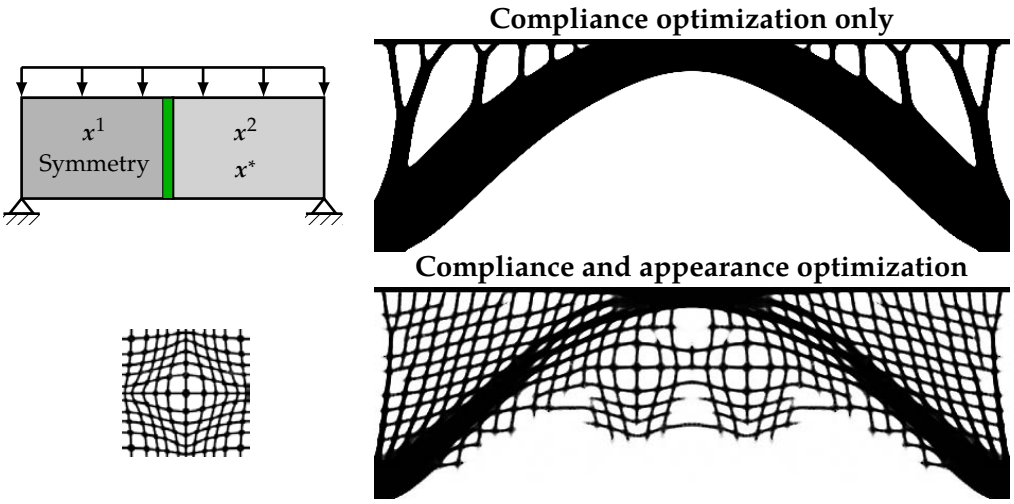


Figure 4.30 – Constrained symmetry, with boundary conditions defined on x^*. Maximum volume is constrained to 45%.

Instead, we propose to optimize structures along several interleaved planes in 3D, as illustrated in Figure 4.31. This is different from independently optimizing 2D shapes: the 3D hexahedral elements at the crossing of several planes are shared, and stresses and appearance propagate across the different planes. The results in Figure 4.31 show how pattern features are able to flow from one plane to another.

4.2.5 Results

Most of our results are obtained by laser cutting from the synthesized shapes. We then assemble objects by gluing several planks together. In most cases, we compute and assemble independent 2D results, under the assumption that forces remain in a plane — which works well in practice for most scenarios. We however also investigated full 3D solutions, as described in Section 4.2.4.

Before presenting our results in more details Section 4.2.5, we describe Section 4.2.5 how we obtain the final curves for laser cutting. We discuss performance in Section 4.2.5 and validation tests in Section 4.2.5.

Contour Extraction

Contours for laser cutting are extracted in a few simple steps. The optimized shape $\overline{x}$ is first thresholded (0.5) to snap values which are between 0–1 to void or solid. In rare cases, this results in the creation of small disconnected components. We filter these by keeping the connected components anchored to attachment points. Disconnected components are further discussed in Section 4.2.5. Finally, the filtered grid is upsampled by bilinear filtering (x2 in our implementation), and paths for laser cutting are extracted along the isovalue 0.5.

Fabricated Objects

We created several objects using our approach. In all cases, the user only specified the attachment points, external forces, target volume and example pattern. The algorithm automatically synthesizes the structure. Thus, many results can be easily produced using a variety of patterns: the algorithm deals with the complex task of generating the intricate details of the final structure. While we only laser cut miniatures, industrial cutters could be employed to fabricate large-scale objects in a variety of materials.

Besides attachment points and external forces, the user may also rely on *passive elements*, which can represent non-designable parts with a fixed density. The

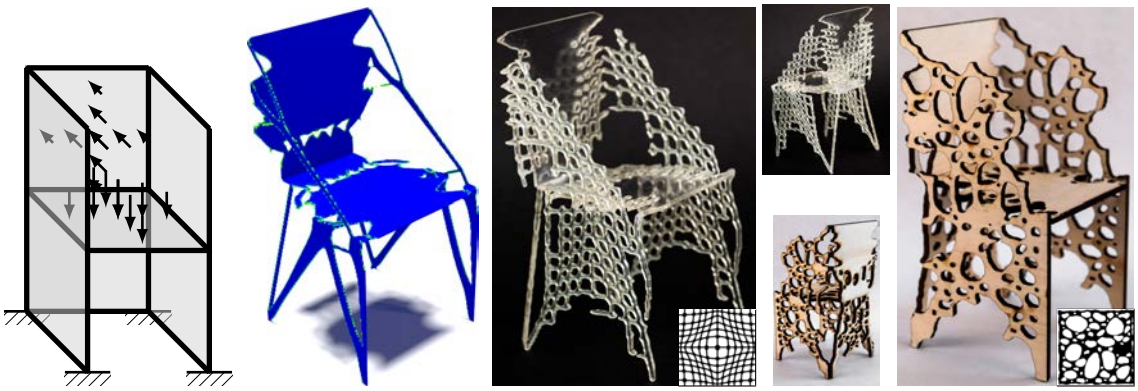


Figure 4.31 – 3D synthesis. **From left to right:** Boundary conditions, result optimized without appearance, two fabricated chairs from our optimized results using different patterns.

inset figure illustrates the use of passive elements to optimize a structure around the SIGGRAPH logo. This is obtained by constraining the value of x_e, that is $x_{min} \leqslant x_e \leqslant x_{max}$. If desired, other passive element properties can be predefined, such as the material volumetric mass density and stiffness.

In Figure 4.32 different bridge sides supporting a road are obtained by changing the loading scenario. A symmetry constraint is also used, but this is optional. Results using different patterns on the same set of conditions are shown Figure 4.23. In Figure 4.33 a set of shelves is produced. They are designed to be fixed on the ground and to support several shelves offset from the attachment point: the weight is entirely supported by the sides. Yet, the structure remains visually similar to the exemplar pattern. In Figure 4.34 we apply the same principle to produce phone stands. Figure 4.35 shows a variety of tables obtained by interleaving three planks. Using different patterns immediately changes the look and feel of the results. The tables are very strong and can support large weights.

Finally, we show in Figure 4.31 3D results where the structures along each plane are optimized jointly in an interleaved 3D problem. This allows the pattern to flow from one plane to another, while in previous results the pattern features could be interrupted between different planks. Also, note how the result optimized without appearance looks much less appealing in 3D, while our results produce intricate, visually interesting details.

Performance

We implemented our approach with Python and use the GCMMA implementation of the NLopt [Johnson 2016] optimization library. We measured the execution time on an Intel® Core™ i7-4770K @ 3.50GHz, 16 GB RAM. Table 4.4 summarizes performance for the main results. On average, in 2D, 75% of the time is spent on

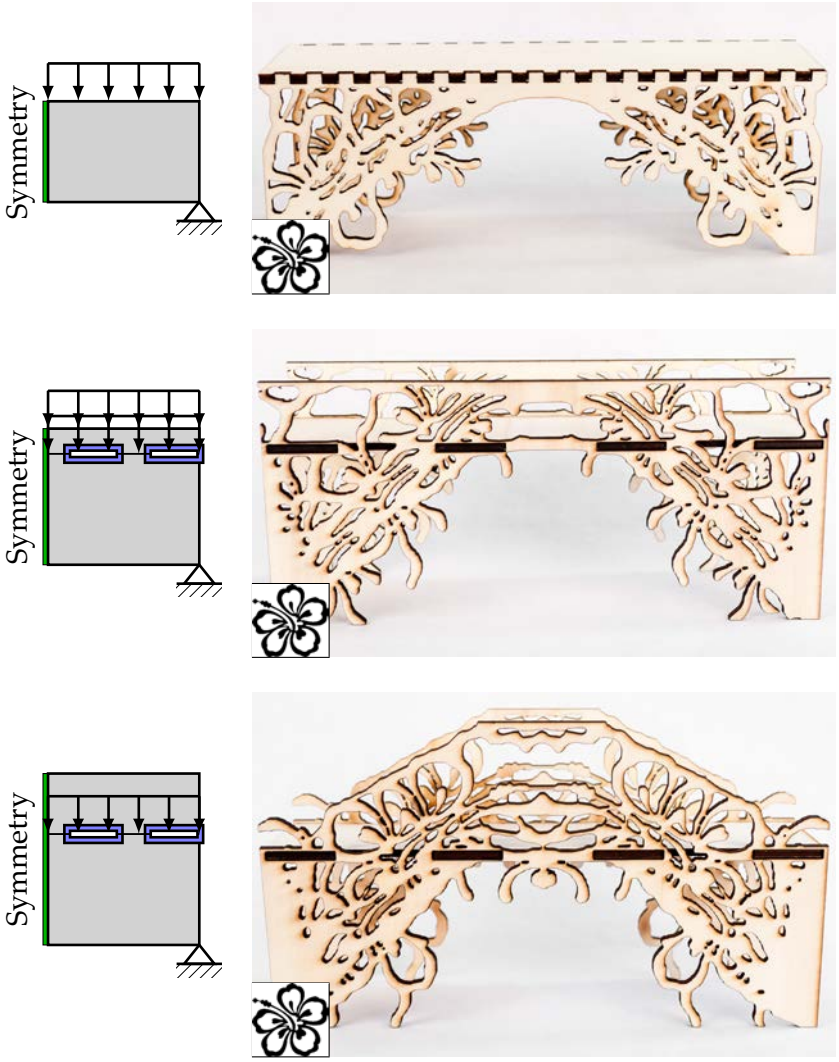


Figure 4.32 – Bridges obtained from the same exemplar using different loading scenarios. Symmetry is used, which is why the boundary conditions are shown for only half of the problem. **Top:** Forces are applied at the top, supporting only the road. **Middle:** Forces are applied below the top for the road, and at the top to create a handrail. **Bottom:** Forces are applied in the middle. Combined with the passive elements this produces an arch.

Figure 4.33 – A set of shelves, meant to be fixed on the ground. Two exemplars are used on the same problem, producing results of very different styles but the same purpose.

Figure 4.34 – Phone stands fabricated from optimized structures using two different exemplar patterns.

the FEM computation, 20% on the appearance gradient computation, and 5% on the GCMMA optimization.

As performance was not our focus, our reference implementation uses a single thread and takes in the order of minutes to converge. Nevertheless, the multi-resolution approach allows the user to preview the result being computed.

As can be seen in Table 4.4, the 3D result of Figure 4.31 takes roughly three times longer to compute than a 2D problem with the same number of elements.

Figure 4.35 – Three tables produced with our system, using three planks and structures optimized independently. The two tables on the left use the same external conditions but different patterns. The right table weights 76 g and is supporting a filled cup of 762 g (10× heavier).

Source	# Elements	Exemplar		Time	# Iterations
Figure 4.23	202 000	Web	240 × 240	8 min 33 s	120
Figure 4.24	62 000	Sponge	200 × 200	1 min 56 s	77
Figure 4.27	150 000	Cells	210 × 210	4 min 20 s	112
Figure 4.29	80 000	Cells	200 × 200	1 min 4 s	56
Figure 4.30	120 000	Grid	160 × 160	5 min 24 s	118
Figure 4.31	114 000	Sponge	104 × 104	14 min 20 s	113
Figure 4.32*	162 000	Flower	176 × 169	5 min 45 s	106
Figure 4.34	230 000	Grid	330 × 330	7 min 38 s	89
Figure 4.36	90 000	Cells	105 × 105	3 min 34 s	119

Table 4.4 – Performance on our main results. The exemplar resolutions are shown as width × height. *Figure 4.32 middle result.

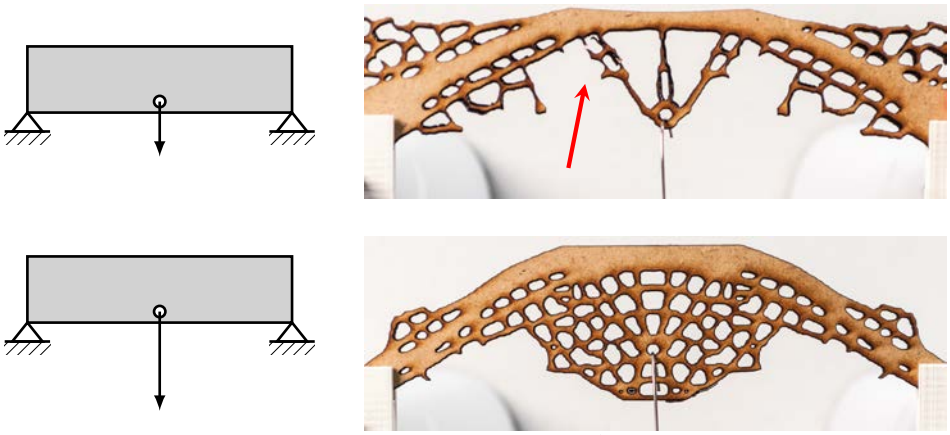


Figure 4.36 – Two structures in MDF wood optimized for different forces but with the same volume constraint. The ratio between the two forces is 10 : 1. The arrow indicates where the top structure starts to rupture when loaded with the heavier weight intended for the bottom structure (please refer to the accompanying video).

Structural Properties

We first verify that the structural properties optimized by our system can be observed after fabrication. We show in Figure 4.36 two bridge structures using the same volume but optimized with a different force in the center. As can be seen, for the case where the force is small the optimizer grows features on the top of the bridge, as this does not violate the compliance constraint. On the contrary, when the force is large the optimizer concentrates material below the arch, for reinforcement. When a heavy load is applied to both, the bridge optimized for a small load collapses, whereas the bridge optimized with the correct force withstands it easily.

Compliance reflects the global rigidity of a shape but does not consider local stresses. Therefore, there is a concern that shapes of low compliance but high local stresses could be produced, resulting in local failures under loads. In practice the results usually exhibit low local stresses, but for a few specific places such as sharp corners. This is illustrated in Figure 4.37 for the classical L-Beam test. Our approach inherits this limitation from compliance-based methods. It is however worth noting that our method produces results having comparable local stresses to those optimized without appearance, as shown in Figure 4.37.

Note that in the field of topology optimization compliance is widely used due to its smoother behavior (see e.g. §2.4 in [Deaton and Grandhi 2014], §6.10 in [Sigmund and Maute 2013]) making it amenable to efficient gradient descent minimization. Optimizing local stresses is still an active research topic [Parıs et al. 2005; Lee et al. 2012a; Holmberg et al. 2013].

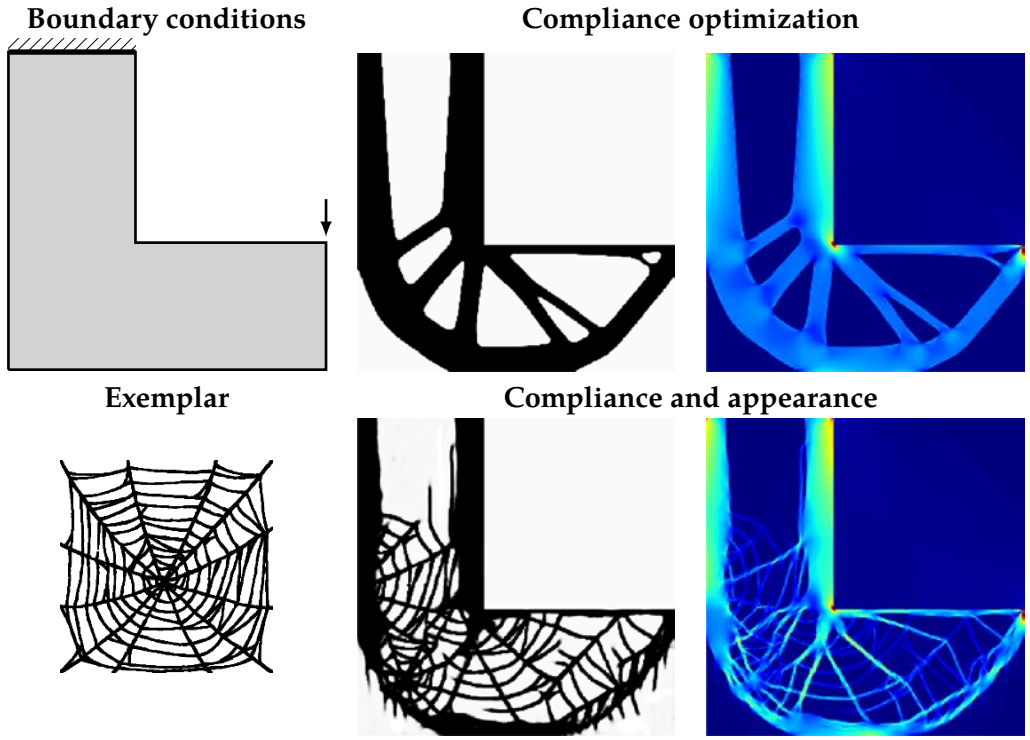
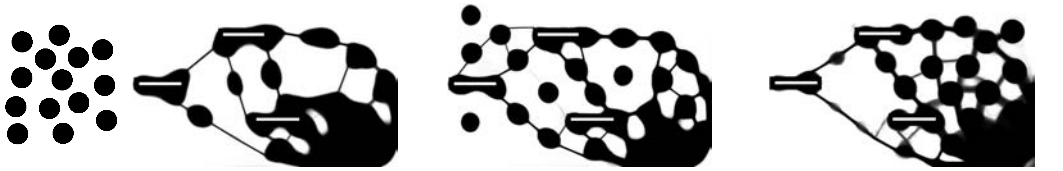


Figure 4.37 – Comparison of Von Mises stresses. $v_{max} = 40\%$, $\alpha = 1.3$. We use the classical L-Beam boundary conditions for stress analysis [Duysinx and Bendsøe 1998] (see top left image). Stress colors use the same normalization of both rows. Our structure (bottom row) has a higher compliance ($\alpha = 1.3$), but local stresses are comparable to the ones found in the result without appearance (top row).

Limitations, Future Work

Connectivity and Convergence. While the SIMP method does not explicitly prevent disconnected components from appearing, they are typically not present in an optimized design. Whenever the compliance constraint is violated, matter is redistributed in weaker regions to reduce the compliance. Combined with the volume constraint this discourages the existence of disconnected components. In all our results, the disconnected elements represent less than 3% of the total volume. However, in cases where the exemplar has many small disconnected components and where the user allows for a high compliance threshold, the appearance objective is free to generate disconnected components (Figure 4.38a).

Self-weight (Section 4.2.4) explicitly penalizes disconnected components: unsupported matter produces high compliance. Therefore, using self-weight encourages well connected shapes (Figure 4.38a). However, combining self-weighting with external forces generally leads to very challenging optimization problems and the



No self-weight, $\alpha = 1.15$ No self-weight, $\alpha = 1.20$ With self-weight, $\alpha = 1.20$

(a) Disconnected exemplar

(b) Non-stochastic exemplar **(c)** Strongly oriented exemplar

Figure 4.38 – Some challenging exemplars.

resulting shapes are not perfectly converged. In addition, self-weight require relaxing the compliance bound (see Figure 4.29). This typically requires the user to test a few different parameters. Strictly enforcing connectivity thus remains an open direction of future work.

Note that there are assumptions about the exemplars that are inherited from texture synthesis. They should exhibit an overall stochastic, homogeneous appearance, and have features roughly the size of the selected neighborhood size. The appearance of organized patterns is not properly reproduced (Figure 4.38b). Strongly oriented patterns make it challenging to create a rigid structure when the features do not align with stress directions (Figure 4.38c).

Thickness Constraints. While we did not impose a minimum length-scale on our designs, previous works exist to control the minimum thickness and hole size in the SIMP framework [Sigmund 2009a; Zhou et al. 2015]. Combining these approaches to ours is left as future work.

4.2.6 Conclusion

Our work enables a novel way to design shapes that are rigid under a set of external conditions. It offers an unprecedented control over appearance through the specification of an exemplar. Rigidity is understood as a constraint affording for a simple and predictable control on the tradeoff between appearance and structural properties.

We envision that expert users will use our technique to quickly produce initial designs serving as a starting point, while non-expert users will explore a large variety of appearances for objects having the same mechanical purpose.

There are several avenues of future work. First, our technique does not scale well to dense 3D problems which are almost impractical due to their large computational cost. Second, some patterns have appearance that complicates the task of achieving a rigid structure, as illustrated in Figure 4.38. While this is currently a limitation we believe that integrating the pattern orientation as an optimization variable will help on the last exemplar, allowing features to align with the local directions of stress. Finally, we are looking forward to explore design tools exploiting our technique.

We hope to bring a novel modeling tool that will empower users — experts or otherwise — to produce shapes that are unique, visually appealing, and yet structurally sound for a given usage scenario.

4.2.7 Additional Results

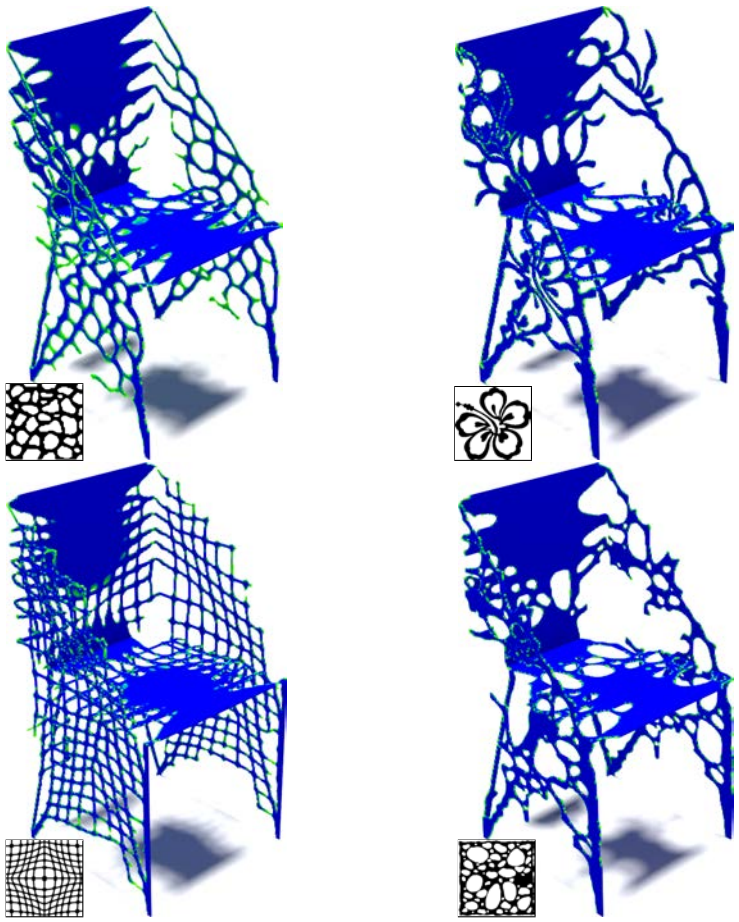


Figure 4.39 – Additional synthesis results, all with a bound of 35% maximum volume.

4.3 Discrete Element Synthesis

Remark. *This section presents preliminary results using a technique developed towards the end of this thesis. Current details are included here for the sake of completeness.*

4.3.1 Introduction

Aggregate geometry, such as a pile of rocks, a box of fruits, a plate of noodles, or furniture assembled from pieces, are ubiquitous in natural and manufactured objects. Due to their importance, aggregate geometry has received significant research focus in rendering [Cook et al. 2007b], modeling [Ma et al. 2011; Landes et al. 2013; Sakurai and Miyata 2014; Roveri et al. 2015], and animation [Kaufman et al. 2008; Hsu and Keyser 2010] to achieve desired appearance and behavior. However, how to design aggregate geometry with desired visual appearance and mechanical properties suitable for manufacturing remains relatively under-explored.

Aggregate geometry often contains sufficient repetitions that can be optimized for both appearance and structure without requiring additional supports such as struts [Stava et al. 2012; Wang et al. 2013; Dumas et al. 2014]. However, existing methods have been applied only to 2D domains, either planes or surfaces [Dumas et al. 2015; Martínez et al. 2015a; Chen et al. 2016], but not 3D volumes which are required for many functional objects. Currently, such 3D aggregates can be procedurally generated [Panetta et al. 2015; Martínez et al. 2016] or manually assembled [Yoshida et al. 2015; Luo et al. 2015], but these methods lack general appearance control. Thus, designing aggregate 3D volumetric objects with desired mechanical structures and visual appearances remains an open problem.

We present a method to automatically generate aggregate volumetric geometry with user controllable visual appearance and mechanical structures. Our system is easy to use, requiring an input exemplar of the desired aggregate patterns and the output domain with desired size, shape, and loading scenario. The output does not require extra support, yet are both feasible during and robust after manufacturing.

Our main idea is to use point samples as design variables for both appearance (as in element texture [Ma et al. 2011]) and structure (as in topology optimization [Bendsøe and Sigmund 2004]). We propose a solver to simultaneously satisfy the following often conflicting goals: faithful reproduction of the input exemplar pattern, observation of the output domain size, shape, and loading scenario, and satisfaction of manufacturing constraints. We adopt the Lagrangian, fluid-like [Overvelde 2012] instead of the Eulerian, grid-based representation to allow more accurate representation and more efficient computation.

Specifically, we use samples to represent geometry [Ma et al. 2011] and optimize their position and orientation in order to minimize the compliance of the system, in a setting inspired by topology optimization techniques [Bendsøe and Sigmund 2004].

Contrary to existing methods in computer graphics (Section 2.3.2) and mechanical engineering (Section 2.5.3), we face additional challenges:

- An efficient 3D simulation need to be performed to allow quick feedback and help the user in the design process.
- The elements need to overlap by a small amount, or be joined together somehow, in a way that does not impair the visual quality of the aggregate structure.
- The elements should be allowed to deform slightly if needed.
- The overall algorithm should be simple enough so that it can be reproduced, integrated or extended for further applications.

The rest of this section is organized as follows. Section 4.3.2 describes in more details our current algorithm for structural optimization using discrete elements. Section 4.3.3 presents some preliminary results, and discusses further possible improvements to the technique.

4.3.2 Overview

Our algorithm takes as input an initial assembly of elements and a description of a mechanical problem: fixed points and external loads applied to the system. The initial configuration can be created manually by a user, or automatically by copying random instances from an input collection of elements.

The elements in the domain are defined by a set of parameters $\boldsymbol{\theta}$, which are the variables used in the optimization. To minimize a structural objective, such as the compliance C, using a gradient-based method, one need to be able to define the compliance C and its gradient in terms of the parameters $\boldsymbol{\theta}$. Building upon density-based topology optimization methods, such as SIMP, the elements are used to define an intermediate density field $\boldsymbol{x}$, which is used to compute the mechanical behavior of the system. A *soft* elastic material is assigned to the regions with density $x = 0$, and a *rigid* solid material is assigned to the regions with density $x = 1$. In between, the material stiffness is interpolated according to x.

To compute the density field $\boldsymbol{x}$ from the parameters $\boldsymbol{\theta}$, each element is represented by a number of point samples [Ma et al. 2011]. In the following, we first discuss our choice of parameterization for rigid elements in 3D. Secondly, we briefly explain how the densities and their gradients are computed from the point samples. Finally, we describe the formula for to the compliance and its derivative in a discrete setting. The final expression of the compliance partial derivative with respect to the element parameters is computed via a a direct application of the chain rule.

Discrete Element Parameterization

Following [Ma et al. 2011], we represent the elements in the domain by point samples. The samples of an element can be selected manually, or they can be computed automatically. We chose the latter option, and used a centroidal Voronoi tessellation [Liu et al. 2009] to sample points regularly inside the input meshes.

If the elements are allowed to deform during the optimization process, the sample positions can be used directly as optimization variables, in combination with additional constraints to prevent excessive deformation. However, if the element shapes must be preserved during the optimization, then the sample positions of each element must be parameterized, e.g. by a translation and a rotation.

Consider an element v in the output domain. Let $\boldsymbol{S}^v$ be a $3 \times m_v$ matrix representing the coordinates of the samples inside v, relative to the reference frame of the element (e.g. the origin set at the centroid of the element). Then, the world space coordinate of the samples in v are given by the relation:

$$\boldsymbol{Y}^v = \boldsymbol{R}^v \boldsymbol{S}^v + \boldsymbol{T}^v \tag{4.23}$$

where $\boldsymbol{T}^v$ and $\boldsymbol{R}^v$ respectively denote the translation matrix and the rotation matrix that parameterizes the element v.

Parameterization of 3D Rotations. Since we seek a gradient-based minimization technique, we need to be able to derive the sample positions $\boldsymbol{Y}^v$ with respect to the element parameters $\boldsymbol{\theta}^v$. While the partial derivatives of $\boldsymbol{Y}^v$ are easy to obtain, there are many different formalisms to express 3D rotations — Euler angles, quaternions, exponential maps, etc. — and not all of them yield a suitable representation for our purposes. Indeed, while 3 parameters suffice to represent a rotation in 3D, certain representations, such as quaternions, use an additional coordinate. While quaternions avoid discontinuities when parameterizing 3D rotations, an additional normalization constraint is required to restrict them to the unit sphere in $\mathbb{R}^4$.

To avoid increasing the problem size with one additional constraint per element, we opt for a three-dimensional parameterization of 3D rotations. One such parameterization is given by the exponential map exp, which provides a surjective mapping from the group of skew-symmetric matrices $\mathfrak{so}(3)$ to the group of 3D rotation matrices $SO(3)$. In addition, the exponential map is known to be a diffeomorphism between the neighborhood of the origin of $\mathfrak{so}(3)$, and the identity matrix in $SO(3)$, which means it behaves well for small rotations.

In practice, we use the formulation given by [Grassia 1998], which computes the exponential map from $\mathfrak{so}(3)$ to $SO(3)$ via an intermediate quaternion representation. The authors provide a C implementation for computing the partial derivatives of the rotation matrix with respect to the exponential map vector in $\mathfrak{so}(3)$. Note that

a compact and direct formula for the derivative of 3D rotations can also be found in [Gallego and Yezzi 2015], however we chose the to use [Grassia 1998] because it implements a Taylor expansion of the exponential map around the origin vector.

Material Densities

To compute the compliance of the system, it is necessary to assign a material density to the current configuration of elements, represented by their point samples. Let s be a point sample in the output domain, and let ρ_s be the material density associated to the sample s. $\rho_s : \mathbb{R}^3 \to \mathbb{R}_+$ is a radial basis function, which depends only on the distance from the sample position $\mathbf{y}_s$, and is parameterized by the sample radius σ_s. σ_s can vary from sample to sample, e.g. depending on how well the ball of center $\mathbf{y}_s$ and radius σ_s locally approximates the shape of the input element. The RBF ρ_s is chosen to have a compact support, e.g. using a Gaussian kernel

$$\rho_s(\mathbf{y}) = A \exp\left(-\frac{\|\mathbf{y} - \mathbf{y}_s\|^2}{2\sigma_s^2}\right) \tag{4.24}$$

where A is a normalization factor. Another possibility is to use a smoothed Heaviside step function

$$\rho_s(\mathbf{y}) = \frac{1}{2} - \frac{1}{2}\tanh\big(k(\|\mathbf{y} - \mathbf{y}_s\| - \sigma_s)\big) \tag{4.25}$$

where k controls the smoothness of the approximation. Note that both function have a rather compact support. Equation (4.24) can be set to 0 when $\|\mathbf{y} - \mathbf{y}_s\| \geqslant 3\sigma_s$. While the support of Equation (4.25) depends on k and σ_s, one should expect it to be close to 0 when $\|\mathbf{y} - \mathbf{y}_s\| \geqslant 2\sigma_s$.

The overall material density $\rho(\mathbf{y})$ is then defined as the max of the densities induced by any sample point $s \in \mathcal{S}$:

$$\rho(\mathbf{y}) = \max_{s \in \mathcal{S}}(\rho_s(\mathbf{y})) \tag{4.26}$$

It is necessary to define the material density as the maximum instead of the sum of individual densities, in order to discourage elements to overlap. For the compliance minimization problem, this means that the samples behave better when they are spread out. However, it should be noted that the max function is technically not differentiable. A common workaround is to resort to a smoothed max formulation, e.g. using a p-norm $\|\cdot\|_p$, with a high $p \geqslant 6$ or 8. The principal drawback of this approach is that the actual density at any point in space depends on the number of samples $|\mathcal{S}|$, as the p-norm inevitably computes some kind of weighted average over the domain. To retrieve a good approximation of the max function, it is necessary

to increase the p exponent when there are a high number of samples, which leads to numerical inaccuracies.

In practice, we have found that simply ignoring this theoretical issue, and retaining a hard max formulation, does not impede the overall gradient computation. Indeed, the only non-differentiable points of the max function are the points which are closest and equidistant from two different samples (assuming σ_s is the same for all samples). In other terms, they are located on the edges of the Voronoi diagram formed by the current distribution of samples.

Discretized Cell Densities. In order to compute the compliance of the system via a finite element method, the material densities are discretized following a regular grid, in our case composed of linear H8 cube elements. Let e denote a cell in the regular grid, and let x_e be its associated material density. The discretized cell density is given by the following relation:

$$x_e = \frac{1}{\text{Vol}(e)} \int_{\mathbf{y} \in e} \rho(\mathbf{y}) \, \mathrm{d}\mathbf{y} \tag{4.27}$$

Equation (4.27) simply means that x_e is defined as the average density ρ over the grid cell e. In practice, the integral eq. (4.27) can be computed either 1) analytically by computing an exact expression of the integral of $\rho(\mathbf{y})$, or 2) numerically by means of a Gaussian quadrature rule, i.e. evaluating $\rho(\mathbf{y})$ at specified points inside the cell. While the expression of $\rho_s(\mathbf{y})$ for a given sample s is integrable analytically, the use of the max function makes it difficult to derive a simple analytic expression of the resulting integral. For this reason, we opted for numerical integration of the expression given in Equation (4.27):

$$x_e = \frac{1}{\text{Vol}(e)} \sum_{i=1}^{N} \omega_i \rho(\mathbf{y}_i^e) \tag{4.28}$$

where the $\mathbf{y}_i^e$ are the evaluation points of the quadrature rule for the cell e, and ω_i are their associated weights.

Compliance and Sensitivities

Given the discrete displacement field $\mathbf{u}$ and external forces $\mathbf{f}$ applied to the system, the compliance of the system is computed as (Section 2.5.1):

$$C(\boldsymbol{x}) = \mathbf{u}^\intercal \mathbf{f} = \sum_e x_e \mathbf{u}_e^\intercal \mathbf{K}_0 \mathbf{u}_e \tag{4.29}$$

where $\mathbf{K}_0$ is the stiffness matrix for the base solid material, and $\mathbf{u}_e$ is the displacement vector associated to the node of the grid cell e.

Following a similar reasoning as in Section 2.5.1, the partial derivative of the compliance can be expressed as

$$\frac{\partial C}{\partial x_e} = -\mathbf{u}_e^\intercal \mathbf{K}_0 \mathbf{u}_e \tag{4.30}$$

Chain Rule. The current pipeline for computing the compliance C from the element parameters can be summarized as follows:

$$\{\boldsymbol{\theta}^v\}_v \rightarrow \{\mathbf{y}_s\}_s \rightarrow \{x_e\}_e \rightarrow C \tag{4.31}$$

Which means that the partial derivative of the compliance with respect to one element parameter can be computed via the chain rule

$$\frac{\partial C}{\partial \boldsymbol{\theta}^v} = \sum_e \frac{\partial C}{\partial x_e}\frac{\partial x_e}{\partial \boldsymbol{\theta}^v} = \sum_e \frac{\partial C}{\partial x_e}\left(\sum_s \frac{\partial x_e}{\partial \mathbf{y}_s}\frac{\partial \mathbf{y}_s}{\partial \boldsymbol{\theta}^v}\right) \tag{4.32}$$

Note that in the above notation, $\frac{\partial x_e}{\partial \mathbf{y}_s}$ is a 1×3 matrix, while $\frac{\partial \mathbf{y}_s}{\partial \boldsymbol{\theta}^v}$ is a $3 \times |\boldsymbol{\theta}^v|$ matrix. While it may seem that the expression of Equation (4.32) is quite complex, a lot of terms in the summation are actually null, since the densities RBF have compact supports, and there is at most one sample contributing to the material density at any given point $\mathbf{y}$ in space — due to the use of the max function in Equation (4.26). Using an acceleration structure such as a k-d tree, and given $\mathbf{u}$ the solution to the FEM equilibrium equation, it is possible to compute Equation (4.32) efficiently.

4.3.3 Results

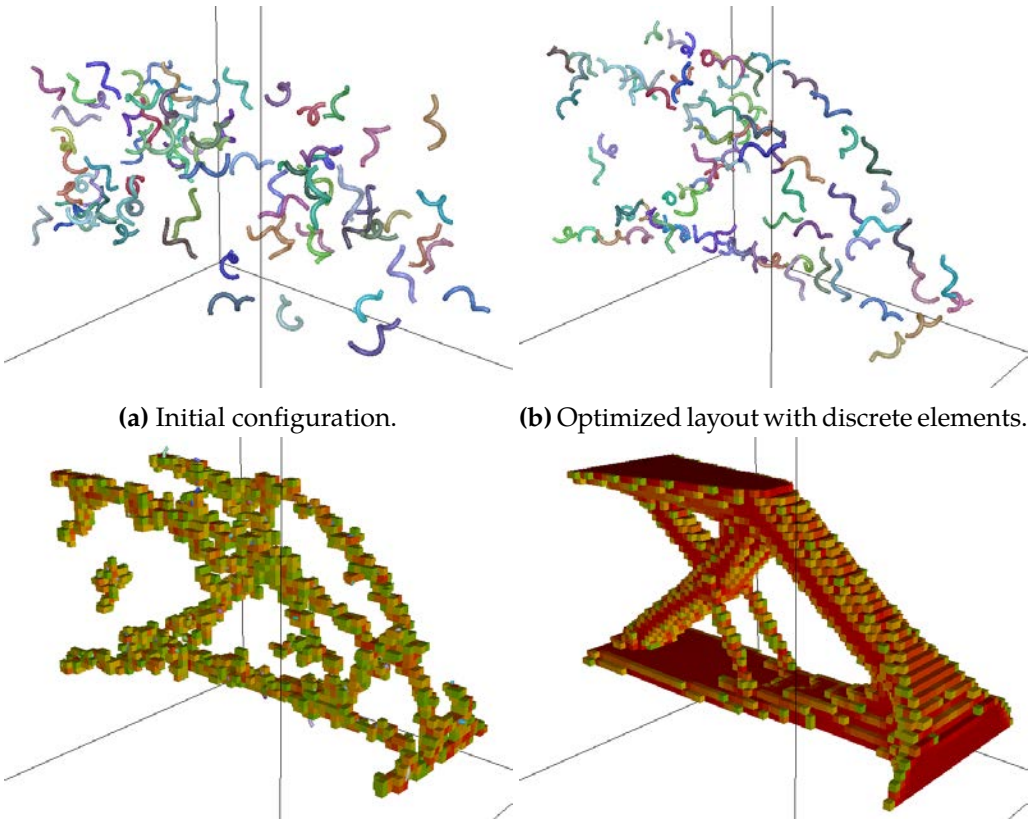
Preliminary results on a 3D cantilever bridge problem are shown in Figure 4.40. The elements are rigid shapes parameterized by a translation and a rotation. When the rotation becomes too high, the element is reparameterized and the current rotation becomes the new reference rotation for the element. The FEM simulation is performed on a regular grid of size $64 \times 32 \times 32$, and the result is obtained in a few minutes on a desktop computer. The FEM equation is solved with a geometric multigrid solver [Amir et al. 2014], which runs on the GPU in OpenCL (the implementation is done within the AmgCL[1] library).

As can be seen, the resolution seems to not be quite sufficient enough to properly capture the element geometries, as the regular grid approximation shown Figure 4.40c

[1] https://github.com/ddemidov/amgcl

actually appears to be quite crude. The FEM simulation is currently the bottleneck of our algorithm, a problem shared by most topology optimization methods. In particular, we are limited by the memory available on a single machine, as the size of the stiffness matrix of the whole system can grow tremendously.

To remedy this situation, we have been investigating adaptive simulation using an octree data structure. Compared to an adaptive volumetric tetrahedral mesh, an octree is both simpler and faster to implement, and it leaves out the possibility for the solver to take advantage of the structured content within the octree.



(a) Initial configuration. **(b)** Optimized layout with discrete elements.

(c) Grid densities associated to (b). **(d)** Equivalent result via the SIMP method.

Figure 4.40 – Preliminary results on a 3D cantilever bridge. Nodes are attached on the left side (top and bottom), and external forces pull the lower-right side downwards.

Adaptive Simulation

To solve the FEM equilibrium equation using an octree data structure instead of a regular grid, some adjustments need to be made. More specifically, instead of solving the discretized equilibrium equation

$$\mathbf{Ku} = \mathbf{f} \tag{4.33}$$

where $\mathbf{u}$ represent the nodal displacements at the nodes of the grid — regular or adaptive —, the nodal displacements at the hanging points (or T-junctions) of the octree need to be interpolated from the free nodes in the octree. If we let $\mathbf{v}$ be the displacement vector for the DOFs of the octree (the non-hanging octree nodes), then we can express the relation between $\mathbf{v}$ and $\mathbf{u}$ as

$$\mathbf{u} = \mathbf{Pv} \tag{4.34}$$

where $\mathbf{P}$ is an interpolation matrix that computes the constrained nodal displacements $\mathbf{u}$ from the nodal DOFs $\mathbf{v}$. It is now possible to retrieve $\mathbf{v}$ and solve eq. (4.33) by pre-multiplying each side with $\mathbf{P}^\intercal$:

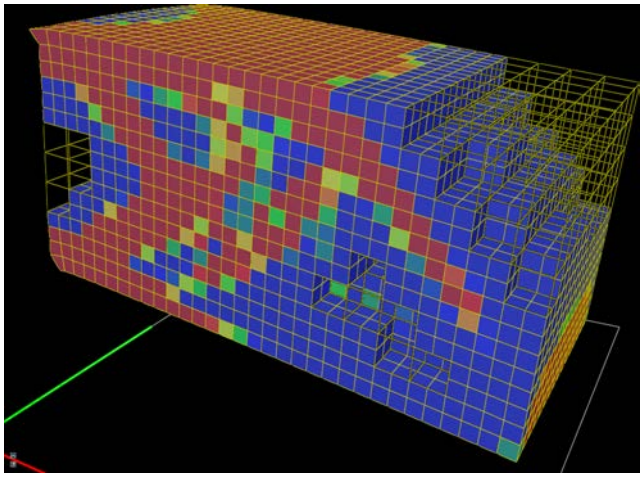
$$\mathbf{P}^\intercal\mathbf{KPv} = \mathbf{P}^\intercal\mathbf{f} \tag{4.35}$$

The result of a FEM simulation on a discretized octree grid is shown in Figure 4.41.

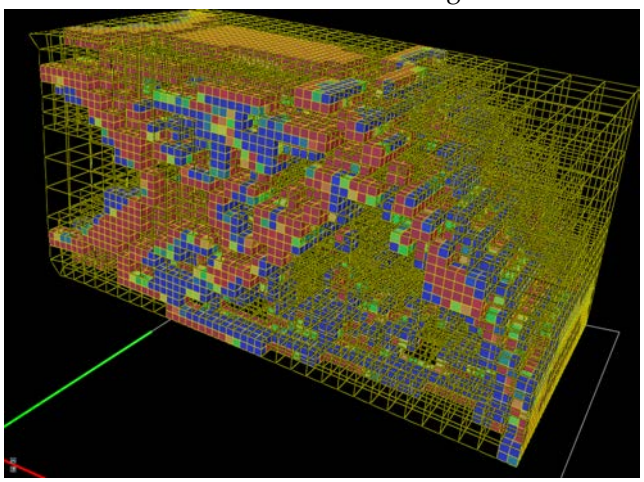
Future Work

There are still areas of this work that need to be further developed. First, the adaptive FEM simulation needs to be combined efficiently with the gradient-descent procedure described in Section 4.3.2. While the subdivision of the octree is almost instantaneous on the CPU, a careful attention needs to be given to the linear solver and preconditioner. Second, flexible elements were not yet investigated in details, and third, a careful control over the element overlap still remains to be exerted. In addition, it is highly possible that a multi-resolution strategy that adapts the radius of influence of individual samples, similar to [Roveri et al. 2015], could help the algorithm converge more quickly to an interesting design, especially in the first stages of the update procedure.

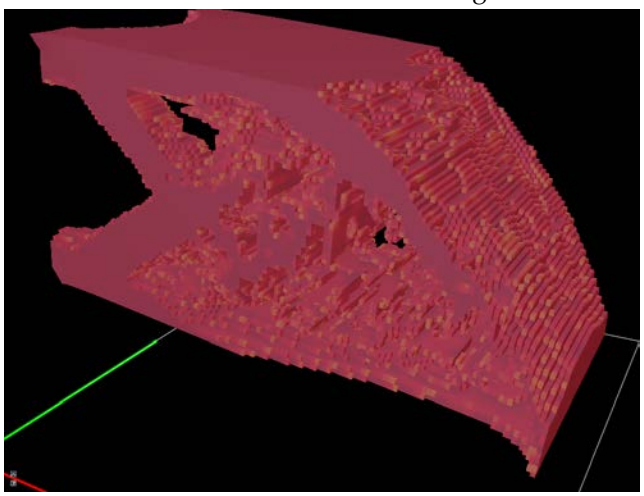
Interior structures that are not visible from outside, due to densely occluded exemplar patterns and/or output domain shapes, can be further simplified [Cook et al. 2007b] or sparsified [Wang et al. 2013] to reduce printing time, save materials, or strengthen structures.



(a) One level of the octree grid.



(b) Another level of the octree grid.



(c) Underlying regular grid at the finest resolution.

Figure 4.41 – Adaptive octree grid and FEM simulation. The FEM equilibrium equation is solved on the octree grid, which as a controlled number of elements.

4.3.4 Conclusion

In this section, we have presented a system for optimizing the layout of discrete elements in 3D in a way that minimizes the compliance of the resulting system. The procedure provides artistic control over the structure by controlling the size of shapes of the aggregated geometries, and the automatic optimizer takes care of the position and orientations of individual elements.

The technique has a lot of potentials, e.g. in terms of aggregating deformable shapes, or stochastic simplification of interior details, that have yet to be explored. We hope that in the future, similar applications will empower the user to explore a variety of designs from a limited set of exemplar shapes, and that intuitive authoring of complex structures can be achieved more efficiently.

4.4 Concurrent Works

In this section, we present other recent advances on shape synthesis for digital fabrication that considers appearance. In particular, we discuss and compare with concurrent works strongly related to the content presented in this chapter.

In [Zhou et al. 2014b], the authors present a method to synthesize a pattern along a curve, from an exemplar, while enforcing connectivity constraints. While this method can be applied along a first direction $\vec{u}$, and applied a second time along a second, orthogonal, direction $\vec{v}$, to give the effect of a 2D synthesis, it is in practice difficult to extend it to form a true 2D pattern synthesis method. Moreover, connectivity is only a geometric constraint, and this work do not cover structural constraints.

After the publication of our methods in [Dumas et al. 2015; Martínez et al. 2015a] (Sections 4.1 and 4.2), a number of approaches dealing with similar problems have emerged. In [Chen et al. 2016], the authors present a method for synthesizing filigree patterns from one or more filigree elements (curvilinear structured elements). This is difficult to achieve with pixel-based methods such as [Dumas et al. 2015; Martínez et al. 2015a], and conversely their method is not well suited to patterns that are not described by basic elements. Their approach starts with an initial dense packing of filigree, then optimizes their position and topology to enforce progressive connections between patterns. A structural analysis step is performed to determine whether more elements should be added to reinforce the resulting pattern on the surface. The optimization is done iteratively, alternating between the two aforementioned steps until no improvements can be made. Their approach work in 2D, and operate on a surface through overlapping patches mapped onto a 3D model. Consequently, as with every method that employs mapping, it will suffer from distortions in high-curvature areas of the model. Another bottleneck of their approach is the structural analysis, which requires extracting a 3D mesh of the surface, to simulate it with shell elements, whereas filigrees themselves can be efficiently represented by their skeleton — a 1D curve.

In a concurrent work of [Chen et al. 2016], Zehnder et al. [2016] developed an interactive editing tool for synthesizing curve pattern. Contrary to [Chen et al. 2016], their method is more focused on interactive editing than automatic by-example synthesis, although they propose an algorithm to create a layout automatically. An advantage of their method is that they operate directly on the surface in 3D, as the curves are modeled as elastic components gliding on a smooth surface, which means the method is less prone — under some assumptions — to distortions artifacts. The structural analysis presented in [Zehnder et al. 2016] is based on an eigenanalysis of the elastic energy of the system. The smallest forces that would incur the largest displacements is computed. Then, the author describe a criteria for suggesting reinforcements, which is very similar to what we present in Section 4.1.4 and [Dumas et al. 2015], as it is based on virtual "soft" edges between points that are geodesically close on the surface. Finally, Chen et al. [2016] provide control over

orientation and scale of the synthesized patterns, while Zehnder et al. [2016] only vary the scale of their elements.

Finally, in a recent work, Schumacher et al. [2016] proposed a method to distribute elements (holes) on a surface while considering both its appearance and structural properties. The appearance objective is formulated as a packing problem, or as a discrete by-example synthesis problem similar to [Ma et al. 2011]. The structural behavior is analyzed using a coarse triangular mesh of the surface (less than 1000 vertices in all examples), and simulated with shell elements. A membrane energy is defined per triangle, and a bending energy is associated to each triangle pairs. Elements are parameterized by the position of their centroid, and to account for their presence in the simulation, a fill ratio α_i is computed for each triangle i, and represents the ratio of its area that is covered by any overlapping element. This fill ratio is used to penalize the membrane and bending energies of shell elements in the simulation. This approach bears strong similarities to the way densities are defined in the SIMP method in topology optimization, which is explained in Section 2.5.1.

In contrast to our objective in Section 4.3, two fundamental differences need to be discussed. First, Schumacher et al. [2016] explicitly prevent element collisions, by adding a repulsive force between neighboring elements. I conjecture that it allows them to use a coarse surface mesh and a simple fill ratio α in the simulation, as detailed element boundaries do not interfere with each others. In contrast, the objective in Section 4.3 is to produce inter-penetrating elements, and thus physical simulation should encourage adjacent elements to slightly overlap. This consideration, plus the fact that the synthesis is performed within a volume, calls for a different type of simulation, which is not a simple extension of [Schumacher et al. 2016].

The second point, which should be easy to adapt, is the choice of the numerical optimizer. Schumacher et al. [2016] seek to minimize a simple weighted sum between appearance and structural energies, whereas in [Martínez et al. 2015a] we advocate the use a more sophisticated constrained optimization scheme, compared to a classical weighted sum. In addition, the L-BFGS-B method employed in [Schumacher et al. 2016] needs to perform an expensive line search to update the design variables. The line search is expensive because each evaluation of the objective function requires recomputing a solution to the equilibrium equation of the system (see previous discussions in Section 2.5.1). For this reason, a solver without line search, such as MMA [Svanberg 1987] or GCMMA [Svanberg 2002], would probably be beneficial to [Schumacher et al. 2016]. Another shortcoming of the L-BFGS-B method, is that it does not handle non-linear constraints directly, and they have to be integrated either in a principled way, e.g. via augmented Lagrangian methods, or in a ad-hoc fashion, e.g. by projecting the design variables onto the constrained subspace after each update. Again, other optimization schemes such as MMA or interior point methods can alleviate this issue, even though at the expense of a slower update at each iteration.

4.5 Discussion and Conclusion

In this chapter, we have presented different ways to meaningfully combine visual and mechanical criteria, as part of an automatic shape optimization procedure. In the following, I summarize and discuss the key ideas introduced by the different techniques developed in the sections above.

Surface Synthesis (Section 4.1). An abstract graph is built as a simple and fast proxy capturing the connectivity of the pattern being synthesized. The size of this abstract graph is independent from the complexity of the pattern in most cases. Only when the pattern has a very intricate topology — e.g. with a high number of disconnected components — will it affect the minimum size of the corresponding abstract graph. In addition, since this abstract graph has a controllable number of vertices, very fast simulations can be achieved by selecting its size accordingly.

Note that certain aspects of this technique can also be improved. First, the meshing algorithm used to reconstruct a volumetric mesh from the abstract graph is very crude. More advanced techniques, such as the tessellation algorithm of [Hart 2008] — used for example in [Panetta et al. 2015] —, would provide more accurate results at the expanse of a higher computation cost. Since the graph itself is already an approximation of the real printed pattern, it is not clear how much accuracy is needed in this surface reconstruction step. Alternatively, using different simulation models, which rely directly on the graph structure, would probably be more adapted.

Second, to correctly assess the accuracy of the simulation model and the approximations made, a more quantitative study should be carried out. Indeed, the comparison presented in Section 4.1.5 is only a qualitative. In addition, mechanical testing equipment often requires standardized test specimens, whereas visual appearance of synthesized results in computer graphics is often better appreciated on more intricate 3D models. In addition to typical models used in computer graphics — such as the Standford Bunny, the Armadillo and the Standford Dragon —, online platforms such as Thingiverse now provides us with a wide variety of models, giving rise to initiatives such as the Thingi10K dataset [Zhou and Jacobson 2016]. Nevertheless, it remains difficult to precisely and easily measure the mechanical behavior of complex manufactured models, especially when they have spatially-varying properties.

Third, the mechanical objective used in the simulation model can be chosen differently. For example, optimizing for local stress (§6.10 in [Sigmund and Maute 2013]), or using modal analysis, to not only detect weak parts of the model [Zhou et al. 2013], but also suggest reinforcements, is an interesting direction of future work. It is interesting to note that other analysis criteria can be used, which do not involve any mechanical simulation. Our geometric criterion in Section 4.1.4 was in fact partly inspired by [Cignoni et al. 2014], where the authors seek to maximize the *isoperimetric number* of a graph to obtain more stable configurations. The isoperimetric number measure the size of *bottlenecks* in a graph. Other measures exist, such as the algebraic

connectivity — defined as the second-smallest eigenvalue of its Laplacian matrix —, or the connectivity — the minimum number of vertices or edges that needs to be remove to disconnect the graph. We note that subsequent work in [Zehnder et al. 2016] also used an idea similar to our geometric criteria, comparing the geodesic distance of endpoints of an edge which are close in Euclidean space. The difference is that they take into account the deformations induced at each endpoints, to evaluate the strain of the edge.

Finally, the texture synthesis method itself has room for improvement. In particular, interactions between texture synthesis techniques based on distance fields [Wu et al. 2014] and level-set methods for structural optimization, is an interesting direction for future work that has yet to be explored.

Joint Optimization (Section 4.2). The key idea in this section is to combine two different energies, one for the appearance, one for the compliance, in a single constrained problem formulation. The problem is solved using a gradient-based optimizer (GCMMA), while giving a meaningful value to the bound on the compliance constraint. The advantage over an iterative alternating scheme is that contradicting contributions to the energy function are handled in a unified manner by a systematic mathematical tool. The result is a design tool that can be controlled via intuitive parameters, in a way that makes sense compared to traditional weighted sums, which tend to mix completely unrelated quantities. A weighted sum of objectives can still achieve visually competitive results, but it is harder to tune, and provides no guarantee on either of the objective function once the process has converged. In the case of multi-criteria optimization, a useful alternative is the search for Pareto optimal configurations [Suresh 2013], which is a direction that we did not pursue.

In practice there are some implementation details that were not discussed in Section 4.2. In particular, it is often beneficial to rescale objective (appearance) and constraint (compliance) functions to a similar range, e.g. between 0 and 100, even if MMA/GCMMA does not perform an explicit weighted sums of these functions. The reason is that the different step size and weights used in the algorithm implementation result in a better numerical behavior if the values taken by the multiple functions are not too disparate. Similar numerical considerations are also discussed in Section 2 of [Svanberg 2007].

Another algorithmic detail that warrants further discussion is the multi-resolution scheme employed in Section 4.2.3. In Section 4.1.3 we were able to perform the surface texture synthesis across 8 or 9 levels of resolution in a hierarchical manner. In contrast, in compliance minimization problems, using too many multi-resolution levels can be detrimental to the computed solution. The reason is that the design at the finest resolution is influenced by the solutions computed at intermediate levels, effectively trapping it into a potentially undesirable local minimum. This is a common problem in SIMP approaches: the solver is very good at converging to an initial guess from a gray initial guess, but moving to a different solution once

the system is already in a black-and-white stage takes a lot of iterations, because the frontier of the design move slowly — see discussion surrounding Figures 1 and 2 in [Sigmund and Maute 2013]. In the case of a multi-resolution strategy, it means that if a small feature could help the hires solution in terms of compliance, but the initial guess inherited from the previous level has "hidden" this small feature — in a void area for example —, then it might be hard for the solver to make it appear later on. Maybe restarting from a "greyed" version of the coarse-level guess could alleviate this issue, but this is left as future work. An example of a multi-resolution scheme for compliance minimization can be found in [Stainko 2005, 2006].

Now, a natural question that arises, is why this is not a problem in texture synthesis? One explanation can be sought in the assumptions made on the synthesized patterns [Lefebvre 2014], which is that they must present a stochastic nature, which should hold across the different resolution levels. Being stuck in a local minimum is usually not an issue in texture synthesis, since the appearance measure is only local, and bad "global" minima are to be avoided. If the input exemplar has a flat color region, a bad global minimum would be an output with a constant color, which is obviously undesirable. In other words, there can be many good local minima which are preferable over global ones, even though they have a "higher" appearance energy, according to the standard definitions. Maybe this calls for a more appropriate definition of the appearance energy — one not subject to this pitfall —, but this discussion is out of the scope of this thesis.

A last fundamental question brought up by the multi-resolution scheme is the choice of filtering process. In Figure 4.26, we used a filter size defined as a fraction of the total problem size. This means that filter covers 4× times more pixels on the coarsest level (0), and 2× more pixels on the intermediate level (1), when compared to the finest level (2). Consequently, the compliance minimizing solution is stable across the levels, albeit the boundaries are "fuzzier" on the coarser levels. This is especially true when using the density filter over the sensitivity filter. Another option is to reduce the filter size when going to the finer levels, knowing that this will change the compliance minimizing solution. Note that more complex filtering schemes may be employed; for example Stainko [2005] uses a combination of two different filtering approaches. The combination of appearance metrics with more involved filtering techniques, along with projection schemes to ensure minimal thickness of the solid and/or void phase [Wang et al. 2011], were not explored during this work, but they are an interesting venue for future work. Note that as more filtering schemes are used (e.g. to ensure minimum thickness), the objective function becomes more challenging for the optimizer (MMA) to minimize, and continuation schemes [Rojas-Labanda and Stolpe 2015a] are required. Interestingly, one could also ask whether the converse formulation — using an appearance metric constraint as a way to filter out checkerboard patterns — could yield a practical alternative to the current filtering methods being used nowadays.

A more practical concern in our current formulation is the lack of local control over the computed solution. In particular, we do not optimize for local stresses, nor do

we put any local constraints on the volume. This results in parts of the design which are completely black, which serves to enforce the global compliance constraint, at the expanse of the appearance metric. See for example the bottom of the bridges in Figure 4.23, or the lower-right corner in the shelf Figure 4.28.

More generally, topology optimization methods provide non-traditional ways to design new shapes. The user has only control over fixed nodes, external forces, but she can define sets of passive elements to restrict the design space. In combination with texture synthesis techniques, and a few selected optimization parameters (neighborhood size, volume constraint, relaxation ratio), this would provide an enriched set of design tools for interactive applications. Note that interactive applications for topology optimization have been previously explored, e.g. in [Aage et al. 2013].

Discrete Elements (Section 4.3). In the last section of this chapter, we showed preliminary results regarding a different kind of structural optimization that considers appearance. Here, the appearance is given by the shape of the building elements, that are to be preserved during the structural optimization step. This yields a different class of texture synthesis technique, labeled *discrete element textures* in the literature [Ma et al. 2011, 2013].

The core idea in Section 4.3 is to represent the complex geometrical shapes by simple point samples, which are then used as proxies to perform the structural optimization. This is in contrast with the more involved techniques presented in Section 2.5.3, and more akin to the approach employed in [Overvelde 2012].

We note that there are two complementary and challenging problems arising with this approach. The first one pertains to the fundamental problem itself, in 2D or 3D, of how to perform the optimization, how to update and connect the elements together. The second difficulty is of a practical order, and relates to performing such an optimization efficiently, especially in 3D. Indeed, the precision required to precisely position neighboring elements requires a fine discretization, which quickly becomes prohibitive in 3D both in terms of time, and memory. To this end, we have started to explore the possibility of using hierarchical formulations, where a higher precision is employed only in ambiguous areas. Note that even with very high resolution, the problem behaves very differently in 2D and in 3D, because elements can connect much more easily in 2D, whereas in 3D they have more freedom to "float around" each others.

Thus, I believe that every practical attempt at solving a topology optimization problem with discrete elements should strive to make it extensible to 3D, even though it is computationally very hard to do so currently. Nevertheless, I believe this original combination to have a lot of potentials in the future, given the increasing capabilities offered by GPUs and parallel computing devices.

Chapter 5

Shape Synthesis with Controlled Elasticity

In the previous chapters we have presented several shape synthesis techniques. In Chapter 3, we were concerned with fabrication constraints, and presented a technique for sparse support generation, aimed primarily at FDM printers. In Chapter 4, the focus was on combining by-example texture synthesis techniques, with methods for structural optimization. In this chapter, we wish to control the elastic properties of parts produced by additive manufacturing technologies. More specifically, in Section 5.1, we seek to control the elastic behavior of printed objects by acting on their micro-structural geometry.

Note that in every chapter of this thesis, different printing constraints were considered to some degree. In Chapter 3, our scaffoldings are inherently stable under static mechanical equilibrium, while the primary objective was to minimize the volume of support material. In Chapter 4, objects are designed according to prescribed load scenario — the user specifies the external forces and fixed points —, while the objective was to match the appearance of an input exemplar, or a given set of input elements. In this chapter, the primary objective is to match the an input scalar field describing the elastic behavior of the given shape, while ensuring the resulting geometry will be printable on the target device.

In order to be suitable for fabrication, techniques that consider the mechanical properties of microstructures need to account for the limited printing resolution, and provide algorithms that take into account the *meso-scale* nature of the printed structures, e.g. [Alexandersen and Lazarov 2015] that does not rely on the homogenization method. Alternatively, synthesis methods that rely on homogenization theory should be highly scalable, to better match the limit theoretical behavior of materials, and prove effective with regards to future technologies, increasing print volumes, and increasing print resolutions.

Our work follows the latter approach, and we tried to propose an original point of view on the matter, combining ideas from procedural texture synthesis techniques while taking into account the mechanical behavior of the synthesized textures. The idea of this project was sparked by two SIGGRAPH 2015 publications [Panetta et al.

2015; Schumacher et al. 2015], which showed the growing interest in microstructure design and fabrication in the computer graphics community. Our original contribution was to take a procedural synthesis approach, as opposed to the regular tiling common to both [Panetta et al. 2015; Schumacher et al. 2015]. The result is a highly scalable method that achieve controllable microstructures synthesis, which are easy to conform to an arbitrary field, even on very large objects. This work was published at SIGGRAPH 2016, in [Martínez et al. 2016], and was done in collaboration with Jonas MARTINEZ and Sylvain LEFEBVRE. The content of our publication is reproduced in Section 5.1 mostly unchanged, apart from the necessary layout adaptations, and an updated Figure 5.6.

Note that methods for procedural texture synthesis have been presented in Section 2.3.2, while efficient methods for generating microstructures — in general without any mechanical considerations — were presented in Section 2.4.1. Aspects regarding mechanical analysis of microstructures, via the homogenization method, as well as other structural optimization techniques, are the subject of Section 2.4.3.

5.1 Procedural Voronoi Foams for Additive Manufacturing

Remark. *Compared to our original publication [Martínez et al. 2016], we have improved certain figures and their explanations, such as Figure 5.3. Readers familiar with our original publication are invited to skip directly to the discussion and conclusion in Section 5.2.*

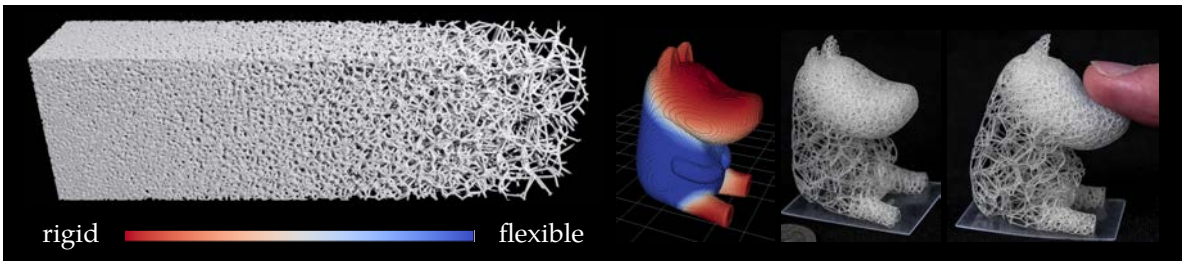


Figure 5.1 – Our method procedurally generates structures with graded material elasticities, which can be directly fabricated. Here the user paints elasticity on a 3D model to create a flexible figurine. Model: Moomin (thing:1173447) by Jeroentjj.

Microstructures at the scale of tens of microns change the physical properties of objects, making them lighter or more flexible. While traditionally difficult to produce, additive manufacturing now lets us physically realize such microstructures at relatively low cost.

In this section we propose to study procedural, aperiodic microstructures inspired by Voronoi open-cell foams. The absence of regularity affords for a simple approach to grade the foam geometry — and thus its mechanical properties — within a target object and its surface. Rather than requiring a global optimization process, the microstructures are directly generated to exhibit a specified elastic behavior. The implicit evaluation is akin to procedural textures in computer graphics, and locally adapts to follow the elasticity field. This allows very detailed structures to be generated in large objects without having to explicitly produce a full representation — mesh or voxels — of the complete object: the structures are added on the fly, just before each object slice is manufactured.

We study the elastic behavior of the microstructures and provide a complete description of the procedure generating them. We explain how to determine the geometric parameters of the microstructures from a target elasticity, and evaluate the result on printed samples. Finally, we apply our approach to the fabrication of objects with spatially varying elasticity, including the implicit modeling of a frame following the object surface and seamlessly connecting to the microstructures.

5.1.1 Introduction

Additive manufacturing enables the fabrication of objects having unprecedented complexity. This capability is often understood in terms of the overall shape of objects. However, it is also possible to fabricate parts filled with *microstructures* — having intricate internal details in the order of tens of microns. The macro-scale mechanical properties of the object are then directly influenced by the geometry of the microstructures. In particular, careful design of the structures affords for parts that are lighter while remaining rigid enough for their intended use. This reduces material usage, shipping and transportation costs. In addition, inner structures can progressively vary within the object and adapt to varying rigidity requirements between regions subject to different stresses.

There are several challenges to achieve these goals. First, the fine scale geometry of the structures has to produce the desired large scale elastic behavior. This often implies formulating challenging global optimizations, either to directly synthesize the fine scale geometry or to fill the shape with precomputed microstructures (see Section 2.4). Second, the structures are very small compared to the size of the objects (e.g. tens of microns versus tens of centimeters). Therefore, the meshes describing the models become quickly prohibitively large, posing important computational challenges for simulation, visualization, and fabrication. Third, the structures have to enforce *fabricability constraints*. The main industrial processes have different requirements depending on whether they locally deposit material (e.g. fused filament fabrication, resin droplets) or whether they locally solidify a bed of material itself acting as a support (e.g. selective laser sintering). In the first case the structures should not present any disconnected parts during fabrication, while in the second case they should not enclose non-solidified material in pockets.

To answer these challenges we draw inspiration from procedural noises in computer graphics, where an infinite amount of content is produced at low, constant memory cost while precisely controlling the statistical properties of the produced noises [Lagae et al. 2009]. This hints at the possibility of generating *procedural*, *stochastic* microstructures that directly exhibit the desired elastic behavior, without further optimization. Our approach explores this idea and defines a procedure to synthesize open-cell foams that enforce fabrication requirements while having precisely controlled elastic properties. The foam parameters can vary spatially to follow gradients of elasticity.

As the microstructures are procedural we only generate their details when needed for fabrication. Typically, evaluation happens at the slice level, just before sintering or curing a layer of material. The microstructures are stochastic and aperiodic in nature. Stochasticity results in an exceptionally good isotropic behavior, and lets us grade the properties without introducing discontinuities along pre-defined boundaries. Aperiodicity removes the need for a global optimization when conforming the structures to a surface.

Contributions.

- The definition of procedural Voronoi foams that can be evaluated very efficiently, have precisely controlled isotropic elastic behavior, and can be spatially graded to produce gradients of elasticity.
- A methodology to derive an inverse mapping, from a target elasticity to the parameters driving the microstructure generation.
- A complete implementation that maps well to stackless, massively parallel architectures.
- Applications to the 3D fabrication of objects filled with procedural Voronoi foams, with proper handling of the outer object frame.

The result is an algorithm that generates microstructures with a prescribed elastic behavior on the fly, during fabrication. No optimization process is required to adapt to a different object or to match a graded elasticity field. There is no limit to the size of the printable objects as the microstructures never fully reside in memory. Therefore, our approach will naturally scale with future technology as the resolution and size of printable object increases.

Scope. Isotropic elastic materials are described by two parameters: Young's modulus and Poisson's ratio. Intuitively the Young's modulus captures how rigid or soft an object is, while the Poisson's ratio captures how one dimension stretches with another. In this work we focus on varying the Young's modulus while preserving the Poisson's ratio of the base material.

For spatially varying elasticity we assume the target scalar field is given as input and do not consider its computation.

5.1.2 Procedural Voronoi Foam Generation

We now introduce our approach for the procedural generation of open-cell foams. We describe the procedural synthesis of the structures in Section 5.1.2 and discuss implementation in Section 5.1.2. We explain how to derive the parameters of the structures from a desired target elastic behavior in Section 5.1.3 and present results and applications in Section 5.1.4.

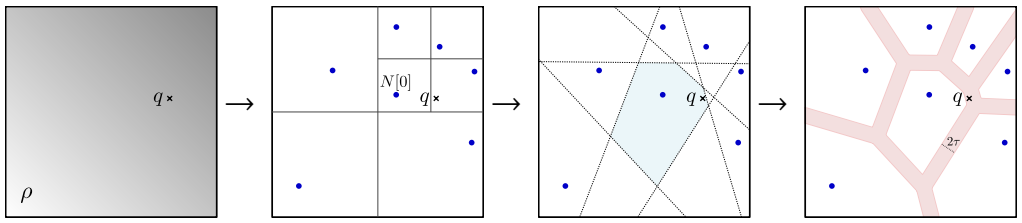
Procedural Synthesis

We seek to define aperiodic *procedural* structures akin to procedural textures in computer graphics. The structure is defined as a function $\mathcal{F} : \mathbb{R}^3 \rightarrow \{0, 1\}$ which is evaluated during display and slicing at every point in space, at the desired resolution.

To have the computational advantages of procedural textures, $\mathcal{F}$ has to follow a number of requirements [Lagae et al. 2010], that we summarize as follows: 1) $\mathcal{F}$ has to evaluate in constant time and constant memory regardless of the point of evaluation. 2) The size of $\mathcal{F}$ — its program and built-in data — has to be independent from the size of the generated content. This is the case of our technique which produces arbitrary large aperiodic content from a constant, small memory footprint (a few hundreds of bytes).

In addition, the structure has to enforce geometric requirements to be printable. First, there should be a minimal number of pockets (holes) enclosing printing material. This advocates for an open-cell structure made of beams along the edges of a cellular structure. Second, there should be no disconnected parts appearing during fabrication. It is easy to see that convex cells enforce this property everywhere but at the boundary (which we discuss in Section 5.1.4). The convex cells of Voronoi diagrams are therefore well suited.

Our procedural Voronoi foams are defined by two parameters: the density ρ of Voronoi seeds per unit volume (seeds/mm^3), and the radius τ of the beams along the edges (mm). Density may vary spatially and is given as a function, i.e. at a given point x the desired density is $\rho(x)$. We assume the variations to be smooth compared to the size of the structure cells.



(a) Density field. **(b)** Gather seeds. **(c)** Eval structure. **(d)** Voronoi edge beams.

Figure 5.2 – 2D overview of Algorithm 14. (a) Input query point q and density field ρ. (b) The seeds that could contribute to the Voronoi cell of $N[0]$ (which is the seed closest to q) are gathered (Algorithm 15). (c) The bisectors of the seed pairs influencing q are computed. (d) Finally, the algorithm checks whether q lies inside a beam of radius τ along the Voronoi edge.

Open-Cell Voronoi Foams. We seek to design a procedural function $\mathcal{F}_{\rho,\tau}$ that produces beams of thickness 2τ along the edges of a Voronoi cell structure having a density ρ. In addition, we would like to allow for ρ to be spatially varying. Our procedural generation is inspired by the seminal work on cellular solid textures by Worley [1996], revised to produce an open-cell structure. We further extend the procedural scheme to afford for spatially varying densities.

Worley defines procedural cellular textures by using pseudo-random sequences to generate seeds in a virtual grid, following a Poisson distribution (not to be confused with a Poisson *disc* distribution). Given an evaluation point q the seed closest to q is determined. The seed id is used to derive a color value at q, for instance coloring each Voronoi cell differently. The set of grid cells that can have an influence is limited to cells neighboring q. This stems from the fact that each cell contains at least one seed — a deviation from a pure Poisson distribution to enable efficiency [Worley 1996] — and, therefore, the closest seed is necessarily within the 2-ring of neighboring cells. This leads to the constant time evaluation property, as the number of considered seed points remains below a constant everywhere in space.

Our algorithm achieves similar properties while generating the edges of a Voronoi open-cell foam. Given an evaluation point q our goal is to write a function returning 0 (empty) if q is not inside a beam of the structure and 1 (solid) otherwise. The pseudo-code of our structure generation is in Algorithm 14, and a graphical overview is in Figure 5.2. It starts by generating all the seeds that could contribute to the definition of the Voronoi cell by calling algorithm GatherSeeds (line 1) — we detail this algorithm later. We next enumerate, for each triplet of seeds $(N[0], N[i], N[j])$, the equations of their line bisector (line 4), as they are potential edges of $\mathrm{Vor}(q)$, the Voronoi cell containing q. For each bisector line, we compute p_L, the point closest to q on the bisector (line 5). If the distance between q and the line is greater than the beam radius τ (line 6), then q is not influenced by this edge. Otherwise, q *might* be inside a beam. To be certain that it is the case, we have to verify that p_L is indeed *on an edge* of the Voronoi cell containing q. To ensure this, we verify that p_L does not strictly belong to another Voronoi cell (lines 8-11). If it does, then p_L is not on an edge of $\mathrm{Vor}(q)$, and we ignore this line equation (line 10). Otherwise, q is in a solid region and we return 1 (solid) line 13. A counterexample, illustrating the importance of this test (lines 8-11) is presented in Figure 5.3.

We next discuss how the seeds are gathered and generated.

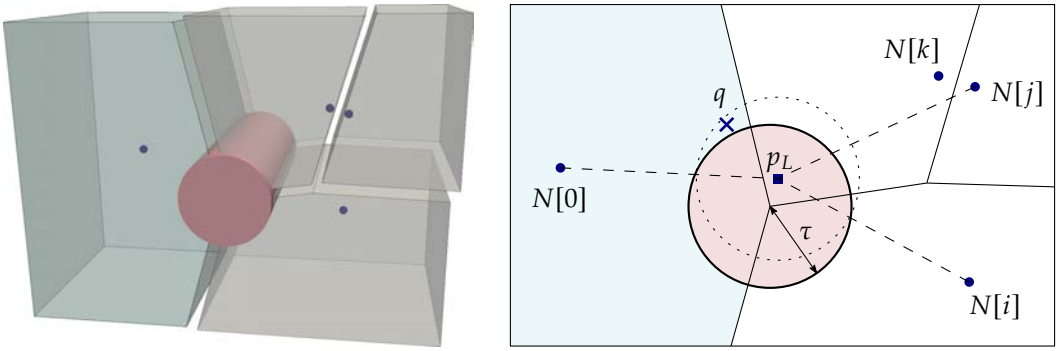


Figure 5.3 – In this counterexample, all the seeds lie on the same 3D plane (cross section shown on the right). The point p_L, which is the closest point to q on the bisector line of ($N[0]$, $N[i]$, and $N[j]$), is at a distance $\tau - \epsilon$ from q, with $\epsilon > 0$. However, p_L does not belong to any edge of the Voronoi cell of $N[0]$ (shaded in blue).

Algorithm 14: EvalStructure $\mathcal{F}_{\rho,\tau}(q)$

Input: Density field ρ, beam radius τ, query point q.
Output: Voxel state $\in \{0, 1\}$.

```
N ← GatherSeeds(q) ; // seeds influencing q
for i ← 1 to |N| do
    for j ← i + 1 to |N| do
        L ← BisectorLineEquation(N[0], N[i], N[j]);
        p_L ← ClosestPointOnLine(q, L);
        if ‖q − p_L‖ ⩽ τ then
            accept ← true;
            for k ← 1 to |N| do
                if ‖N[0] − p_L‖ > ‖N[k] − p_L‖ then
                    accept ← false;
                    break
            if accept then
                return 1
return 0
```

Gathering Seeds. Algorithm 14 requires all seeds that can influence the result at q. If some required seeds were to be missed, the produced structure could fail to print or break. However, we only need to be conservative: as long as we have a superset of the required seeds, the algorithm will produce the correct result.

We generate seeds in a grid, and we guarantee that all grid cells receive at least one seed. This bounds the number of grid cells that we have to consider. As explained in Figure 5.4, the Voronoi cell of a seed cannot be influenced beyond a 2-ring of neighbors.

Algorithm 15 gives the pseudo-code for gathering the seeds around q. The evaluation point q might belong to the Voronoi cell of any of the seeds within (at most) a 2-ring radius. We therefore first search for the seed closest to q, and then gather seeds with a 2-ring around the closest seed. SubdivideCell produces at least one seed per grid cell — possibly more with spatially varying density. We detail this algorithm next.

Seed Generation and Density Control. We now describe SubdivideCell. It generates seeds in each cell, at least one and possibly more by subdividing to locally adapt to density. Conceptually our technique is based on a primal subdivision of the coarsest density grid, where each parent cell is split regularly in eight children. It is important to recall, however, that the grids are never stored: all computations happen implicitly and on the fly. We refer to the first level of grid cells as the *coarse* grid cells (between 2 mm and 5 mm in our implementation). Another important

Algorithm 15: GATHERSEEDS(ρ, q)

Input: Density field ρ, query point q.
Output: Set of seeds possibly influencing q.

```
1  N ← ∅ ;
2  visited ← ∅ ;
3  closest ← ∞ ; // closest seed (initially at infinity)
4  c_q ← GRIDCELLENCLOSING(q) ;
5  for cell ∈ TWORINGNEIGHBORHOOD(c_q) do
6      visited ← visited ∪ {cell};
7      seeds ← SUBDIVIDECELL(cell, ρ);
8      N ← N ∪ seeds;
9      for s ∈ seeds do
10         if ‖s − q‖ < ‖closest − q‖ then
11             closest ← s;
12 c_s ← GRIDCELLENCLOSING(closest) ;
13 for cell ∈ TWORINGNEIGHBORHOOD(c_s) \ visited do
14     N ← N ∪ SUBDIVIDECELL(cell, ρ);
15 return N
```

design goal of our subdivision process is to avoid bias: The statistics of the point distributions remain constant — up to a scaling factor — at all density levels.

The pseudo-code is given in Algorithm 16 for the general case of a spatially varying density field. Given a grid cell of size l and center c, we compute the number of seeds it has to contain as $l^3 \times \rho(c)$ (line 2, cell volume times density). We evaluate the density at the cell center, but more elaborate schemes could be used (e.g. multi-point evaluation, or pre-integration of ρ in a summed area table). The field ρ is clamped to a minimum value to ensure that the coarsest cells always receive at least one seed. When the target number of seeds in a cell is above 2^3 — there is more than exactly one seed per subdivision child of the current cell — we recurse and subdivide the cell (line 13). Otherwise, we randomly select $n = \lfloor l^3 \times \rho(c) \rfloor$ distinct children and draw exactly one seed in each (lines 4-7). We take into account the remaining fraction $f = t - n$ by drawing an additional sample in a next child cell, with probability f (lines 8-10). Note that all random number generators are pseudo-random sequences seeded by the grid cell coordinate. Therefore, for a same initial grid cell, the exact same set of seeds is produced.

The pseudo-code in Algorithm 16 cannot be implemented directly on massively parallel architectures (GPUs) due to the recursive calls. We present in the supplemental material a stackless iterative version that maps well to massively parallel processors.

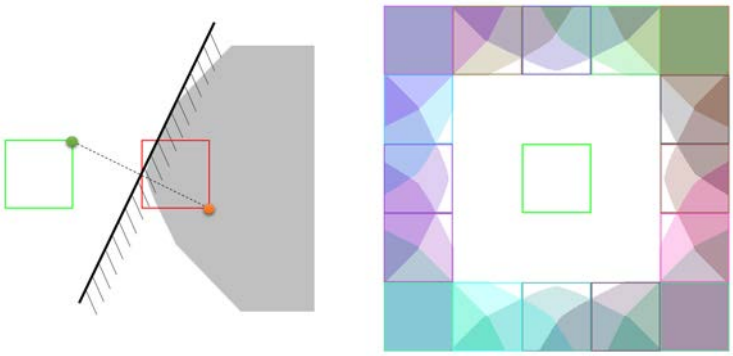


Figure 5.4 – Left: We consider the Voronoi cell of a seed located anywhere in the green square, and how it is influenced by another seed in the red square. The red grid square is within the second ring of neighbors of the green grid square. Each pair of seeds in the green/red squares defines a possible bisector, which might be a face of the Voronoi cell of the green seed. The opposite half-space cannot belong to the Voronoi cell. This is illustrated for a single pair of seeds in the figure (green/red dots). We define the *shadow* of the red square as the intersection of all the half-spaces from all possible pairs of green/red seeds. The shadow is shown in gray ; it represents the region of space that *cannot* possibly be part of the Voronoi cell of the green seed: regardless of the position of the seeds, the shadow is always cutout by the bisectors. **Right:** The shadows of all grid squares in the 2-ring completely cover the space beyond. Therefore, no seed outside of the 2-ring can have an influence on the Voronoi cell of the green seed.

Beam Radii. The radii of the beams are directly controlled by τ during evaluation (Algorithm 14, line 6). While this value may also vary spatially ($\tau(q)$) we only vary density in our approach (see Section 5.1.3).

We next discuss how to select the parameters to reach a target elastic behavior.

Implementation

We implement our technique in an image-based slicer, which either directly sends images to the printer (SLA) or extracts contours (SLS). We slice at a resolution between 10 µm and 50 µm per pixel.

The procedural foam is implemented as an OpenCL kernel processing all pixels of a slice in parallel on a GPU. In addition, we implement a supersampling procedure to obtain gray-scale exposure levels on SLA processes. This is done by performing supersampling around the evaluation point q in a small loop around lines 6-13 of Algorithm 14, counting the number of times $q + \epsilon$ lies within the beam.

Algorithm 16: SUBDIVIDECELL

Input: A cell to subdivide, a density field ρ.
Output: A set of seeds, with a density driven by ρ.

$N \leftarrow \emptyset$;
$t \leftarrow \text{currentCell.length}^3 \times \rho(\text{currentCell.center})$; // target number of seeds
if $t \leqslant 2^3$ **then**
    I = RANDOMPERMUTATION(SUBCELLS(currentCell));
    $n_{\min} = \lfloor t \rfloor$; // minimum number of samples to draw
    for $i \leftarrow 0$ **to** $n_{\min} - 1$ **do**
        $N \leftarrow N \cup \{\text{RANDOMSAMPLEINSUBCELL}(I[i])\}$;
    $p \leftarrow \text{RANDOMFLOAT}(0, 1)$;
    if $p \leqslant (t - n_{\min})$ **then**
        $N \leftarrow N \cup \{\text{RANDOMSAMPLEINSUBCELL}(I[n_{\min}])\}$;
else
    for subcell $\in$ currentCell **do**
        $N \leftarrow N \cup$ SUBDIVIDECELL(subcell, ρ);
return N

Exact beam contours could be extracted by *locally* constructing the explicit geometry of the Voronoi cells around q in the manner of Algorithm 14. We found this approach less convenient than our implicit description that fits very well existing massively parallel architectures, and avoids having to union beam geometries explicitly.

5.1.3 Foams With Controlled Elasticity

We now consider the problem of selecting the microstructure parameters (ρ, τ) to achieve a target elastic behavior. This is done by statistical analysis of the homogenized elasticity tensor of $\mathcal{F}_{\rho,\tau}$, for different parameters.

Analysis Using Homogenization

For a choice of (ρ, τ) we produce a periodic version of the structure in a base volume, and apply homogenization to compute the elastic tensor of the corresponding periodic media. By doing this for many choices of (ρ, τ) we reconstruct the underlying relationship between ρ, τ and the elastic properties (Young's modulus, Poisson's ratio). The graph resulting from this analysis is shown in Figure 5.5. It is obtained for a material with a unit Young's modulus (= 1) and a Poisson's ratio $\nu = 0.3$ which matches most plastics. Only changing the Poisson's ratio requires re-running the simulation as results scale linearly with Young's modulus.

We perform homogenization on a grid of hexahedral elements (see supplemental material in Section 5.1.5 for details). The size of the elements is chosen to properly capture the beams; our experiments showed that using half the diameter gives stable results. The stiffness of each element is computed by calling our implicit procedural function with super-sampling, which returns the volume of structure intersected by the element.

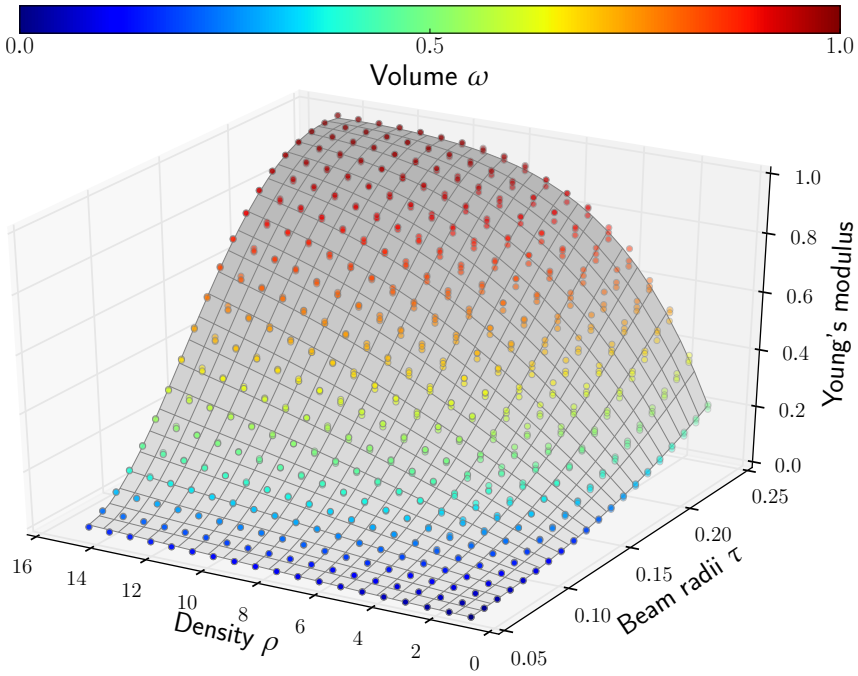


Figure 5.5 – Young's modulus as a function of radius and density. Each dot is the result of homogenization for a given (ρ, τ). For each (ρ, τ) we generate multiple instances of foams using different random seeds. As can be seen the dots tightly cluster, indicating that randomness has little impact on the elastic properties. The maximum deviation after the fitting is 3.3% of the Young's modulus. The data points are used to fit a polynomial function (see Section 5.1.3). For clarity, the graph shows a subset of our data that extends to higher densities for lower beam radii.

Homogenization of Aperiodic Foams. With this approach we are making an important assumption. Our structures are *not* periodic: they form an aperiodic foam of constant density in space. We therefore assume that the overall behavior of the aperiodic foam is similar to the periodic behavior of a *sufficiently large base volume*. To determine the size of the base volume we rely on the expected isotropic behavior of irregular open-cell foams [Luxner et al. 2007]. We perform homogenization for volumes of increasing spatial extent and consider the deviation of the computed tensor from a perfectly isotropic tensor.

The homogenized tensor is a full symmetric tensor composed of 21 independent variables. We approximate the full tensor with the tensor of an isotropic material and consider the residual error. The tensor of an isotropic material is expressed from only two independent variables — Young's modulus and Poisson's ratio — and we denote it as $C^I(E, \nu)$. Its expression is given in the supplemental material.

Similarly to [Schumacher et al. 2015] we approximate the homogenized elasticity tensor C^H by minimizing

$$\begin{aligned} \xi(C^H) = \min_{E,\nu} \|C^I(E,\nu) - C^H\|_F^2 \\ -1 \leqslant \nu \leqslant 0.5 \\ 0 \leqslant E \leqslant E_M \end{aligned} \tag{5.1}$$

where E_M is the upper bound Young's modulus of the solid base material. While the expression of C^I is nonlinear with respect to E, ν, it becomes linear when expressed in terms of the Lamé parameters, and thus it can be computed via least square minimization.

The approximation error measures whether homogenization was able to properly capture the isotropic behavior of the structure. Figure 5.6 reports the error in isotropy for various grid sizes. In practice, we find that a volume having at least $60 \times 60 \times 60$ hexahedral elements provides a stable result across a wide range of beam radii.

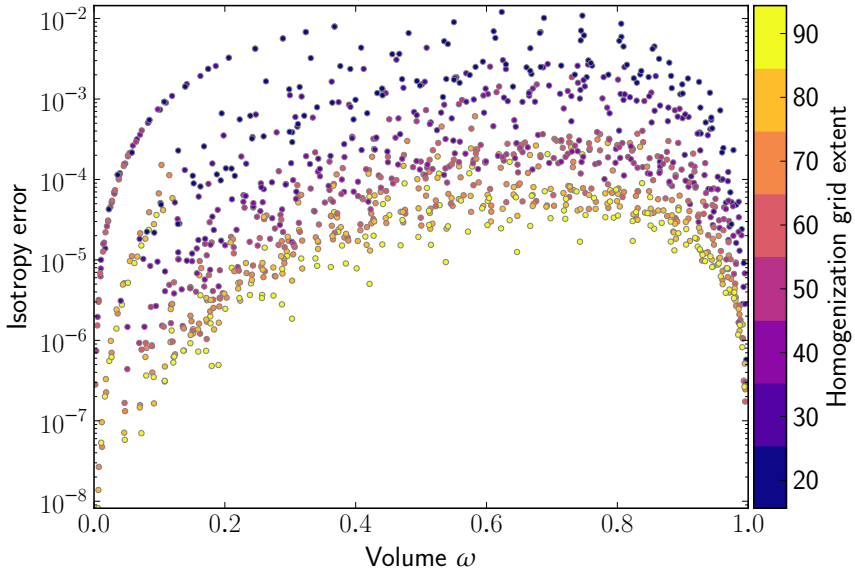


Figure 5.6 – Relative volume — volume of the structure over simulated volume — and error $\xi(C^H)$ of the approximation of the ideal isotropic tensor (i.e. divergence from isotropy). As the spatial extent of the simulated volume grows the tensor tends towards the ideal isotropic tensor.

Collecting Data Points. In a one-time precomputation we generate a large number of pairs (ρ, τ). We uniformly sample ρ and τ to cover the range that can be fabricated (see Section 5.1.3). For each selection of (ρ, τ), we generate a number of different structure realizations by varying the global random seed. We compute the Young's modulus and Poisson's ratio obtained by applying homogenization on each unit volume. This gives us a large number of data points characterizing the elastic behavior of the structures, as shown in Figure 5.5.

We then fit a polynomial function $\mathcal{P}$ on the experimental results to correlate the density, beam radius, and Young's modulus. We optimize for the coefficients of a degree 4 polynomial by least square fitting, minimizing $\sum_i \left(\mathcal{P}(\rho_i, \tau_i) - E_i\right)^2$. The resulting polynomial fit is shown in Figure 5.5.

Deriving Parameters for a Target Elasticity

We now describe how to select ρ and τ to achieve a target Young's modulus. As can be seen in Figure 5.5, for a given target elasticity multiple choices of ρ and τ are possible: the full isocurve where $\mathcal{P}(\rho, \tau) = E_{\text{target}}$. Our preferred strategy to select the values of ρ and τ is to obtain structures that are as dense as possible, to remain close to the limit behavior computed by homogenization. The limiting factor is the minimal printable beam radius, which we denote as $\tau_{\min}$. To maximize density, we fix $\tau = \tau_{\min}$. As densities increases with a fixed radius, we reach a point where the block of matter is full. There is no need to further increase density, which leads to a density upper bound $\rho_{\max} = \frac{1}{(2\tau_{\min})^3}$. The resulting curve is extracted from the polynomial shown in Figure 5.5 by intersecting it with the plane $\tau = \tau_{\min}$ — in practice we use $\tau_{\min} \in [0.2, 0.3]$ mm depending on the printer.

For high densities, just before reaching a full block of matter, small pockets start to appear. This is due to the edges of the Voronoi cells merging together. As measured in Figure 5.7 the volume captured by pockets never exceeds 4%, and is negligible for relative volumes below 0.7.

Other strategies could be developed. For instance, for aesthetic purposes a user could use thicker beams and a lower density to reveal the structures at the same elasticity.

Elastic Behavior: Properties and Analysis

The collection of data points we produced for the parameter fitting lets us fully characterize our procedural foams. We now compare our results to the open-cell foams literature and verify our microstructures behave as expected.

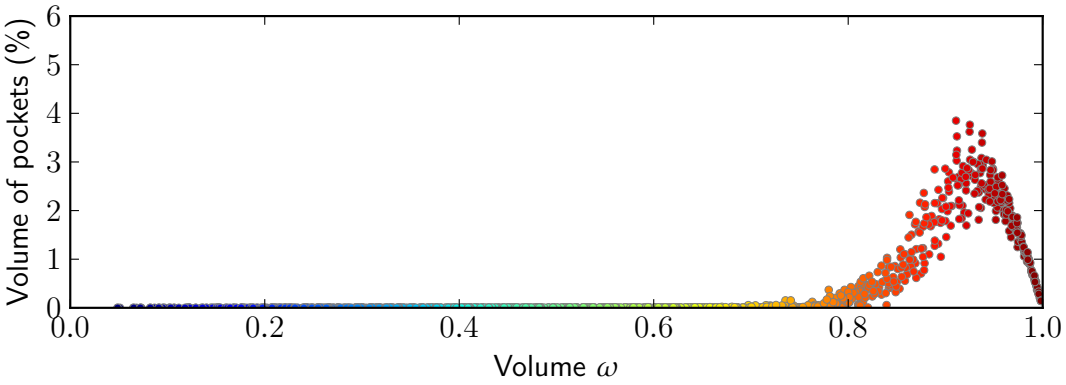


Figure 5.7 – Relative volume ω and volume of pockets (%). Pockets have a negligible impact under a relative volume of 0.7 and go up to 4% of the full volume around 0.9 relative volume. Thanks to the quasi-linear relationship between relative volume and Young's modulus, we can expect at most a similar error on the elastic behavior.

Elasticity. The linear elastic response of *low density* ($0.04 < \omega < 0.5$) open-cell foams is described by the following model [Gibson and Ashby 1997]:

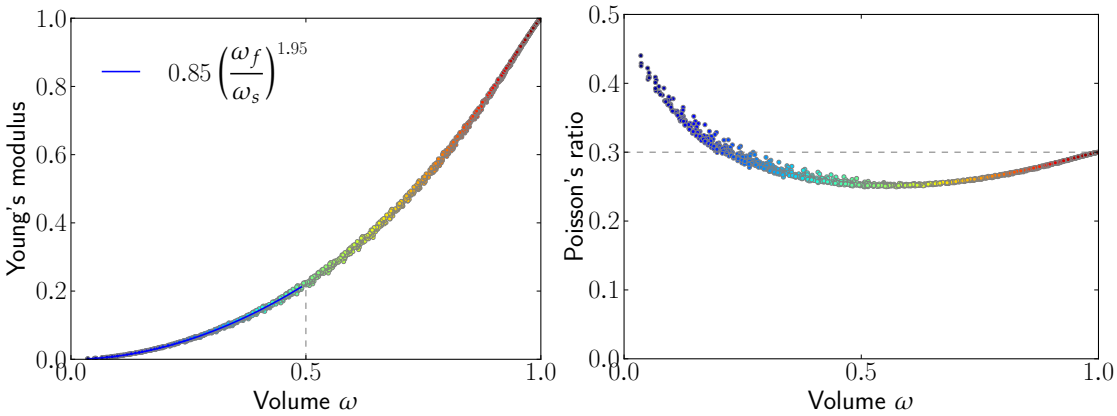
$$\frac{E_f}{E_s} = C\left(\frac{\omega_f}{\omega_s}\right)^n \tag{5.2}$$

where the subscript f denotes the foam and s denotes a solid block of base material. $\frac{\omega_f}{\omega_s}$ defines the relative volume ω. The constants C and n change depending on the foam family [Roberts and Garboczi 2002].

Figure 5.8a shows the relationship between the relative volume and the Young's modulus E of our procedural foams. We fit the model for low density foams on the range $0.04 < \omega < 0.5$, using $C = 0.85$ and $n = 1.95$. As can be observed, our data fits this model very well.

Figure 5.8b shows the Poisson's ratio of our structures, and as can be seen it remains stable. While different materials will give different values of Poisson's ratio, it always remains stable around the Poisson's ratio of the base material as observed by Gibson and Ashby [1997] on open-cell foams.

Randomization. Our procedural foams are based on a (pseudo) random process. Therefore, it is important to consider whether different realization using the same base parameters ρ, τ produce a consistent Young's modulus. Figure 5.5 reveals this as for each pair (ρ, τ) we produced three different realizations. Close inspection reveals that the different realizations do not perfectly match. The largest deviation to the fitted polynomial is 3.3% of the Young's modulus and therefore is negligible. This is in agreement with the literature where it was observed that randomization



(a) Young's modulus. **(b)** Poisson's ratio.

Figure 5.8 – Relative Young's modulus and Poisson's ratio. The base material has a Poisson's ratio of 0.3. The Poisson's ratio tends to 0.5 for low densities, and to the base material Poisson's ratio for high densities.

has little influence on the linear elastic properties of open-cell foams [Van Der Burg et al. 1997].

Regularity. Studies of naturally occurring foams often discuss the *regularity* of the foam, which in our case measures the minimum distance between Voronoi seeds in a bounded domain using the same number of seeds. A Poisson disc distribution provides the most regular cases, while a random point process is the less regular. Our procedural foams lie in-between.

Our seed generation draws at least one sample per grid cell. This is known to be a crude — yet very efficient — approximation of a Poisson disc distribution [Cook 1986]. The approximation is less evenly spaced than a high-quality Poisson *disc* distribution. However, less regular foams exhibit a better isotropy [Luxner et al. 2007], which is also desirable. Therefore, we believe that our jittered grid approximation is a good compromise between computational efficiency and elastic properties, while allowing for graded densities.

5.1.4 Applications and Results

We now experimentally challenge the behavior of our foams. In Section 5.1.4 we compare their isotropy to recent tile-based techniques, in Section 5.1.4 we measure actual printed samples, and in Section 5.1.4 we describe a complete application to fill 3D models with graded elasticity.

Isotropic Behavior Versus Periodic Tiles

Figure 5.9 compares the isotropy of the tensors computed by homogenization using tile-based methods. The method of Panetta et al. [2015] explicitly constructs isotropic tiles. When discretized in our homogenization process they provide close to perfect isotropy; the measured error is due to the limited numerical precision. At a same scale and resolution our structures exhibit lower isotropy; however, performing homogenization with increasingly larger volumes reveals that the residual quickly decreases (see Figure 5.6). This shows that filling larger volumes with procedural foams improves isotropy — a property that stems from the aperiodic stochastic nature of Voronoi foams. Figure 5.9 also shows a tile with a less isotropic behavior (manually designed to match [Schumacher et al. 2015]).

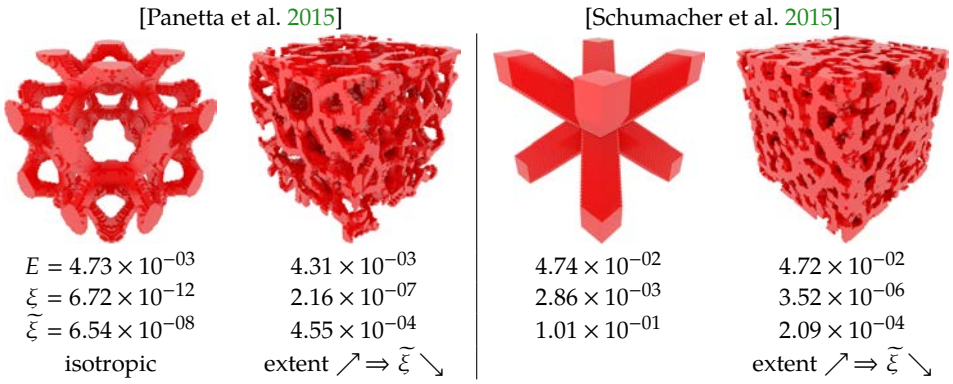


Figure 5.9 – Relative distance $\widetilde{\xi}$ to an ideal isotropic tensor, for a fixed voxel size and a resolution of $80 \times 80 \times 80$. Both pairs have a similar Young's modulus. The small isotropy error of the tile from [Panetta et al. 2015] (left) is due to limited numerical precision and decreases with finer discretization. As larger volumes are simulated, the isotropy of our structures quickly improves (see also Figure 5.6).

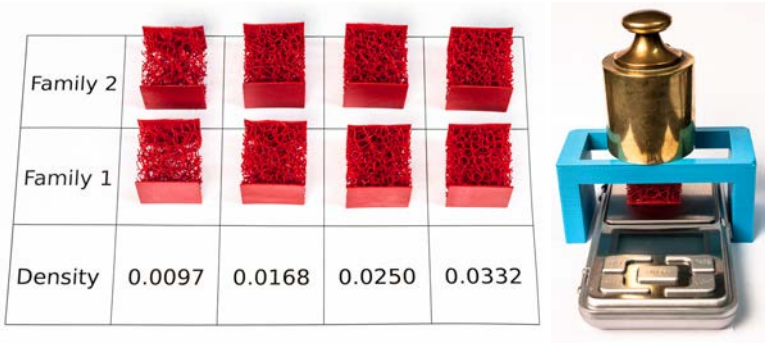
Experimental Results on Printed Samples

We verify the predicted elastic behavior of our foams on printed samples. We print two families of four samples with varying Young's modulus. Each family uses a different random seed. All samples are printed on a *B9 Creator V1.2* using red-cherry material. The two families are visible in Figure 5.10.

We put each sample on a high precision scale and impose to each a same small displacement. We then measure the weight applied to the sample onto the scale (canceling out the structure weight). Due to the limited maximum measurable weight of the scale we consider samples with relatively close Young's modulus. The results are reported in Figure 5.10. We obtain a good correspondence between the measured data and the prediction; the average deviation of the Young's modulus

from linear correlation (dashed line in Figure) is 0.001434, that is less than 0.2% absolute error, and 25% relative error.

For reproducibility please note that the SLA print process, the subsequent cleanup and the material curing all have significant variabilities. We noticed on a few prints (2 out of 16) an incorrect final weight — we discarded these samples. We also noticed that the elastic behavior significantly changes in the first few hours after the print completed. Similarly to [Panetta et al. 2015] we wait for 24 hours before taking measurements.



Family	Target Young's modulus	Density by polynomial	Object weight (g)	Measured weight (g)
1	0.0045	0.0097	1.47	35.9
	0.006 75	0.0168	1.66	57.98
	0.009	0.025	1.82	100.82
	0.011 25	0.0332	1.91	147.36
2	0.0045	0.0097	1.42	32.32
	0.006 75	0.0168	1.67	60.06
	0.009	0.025	1.78	99.2
	0.011 25	0.0332	2.07	153.86

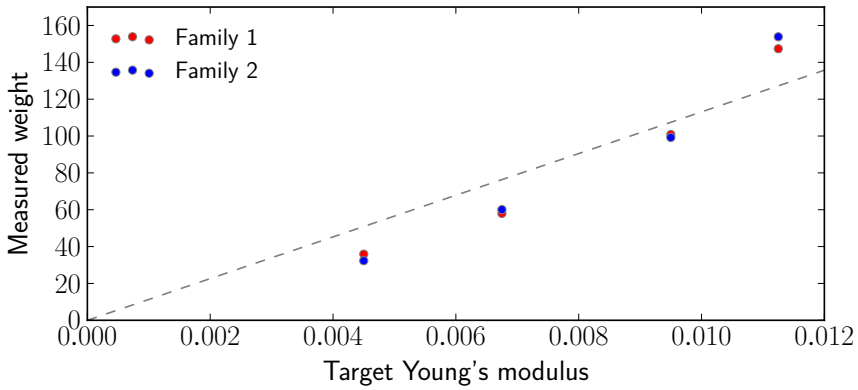


Figure 5.10 – Compression tests on printed samples. The measured weight is expected to be linearly correlated with the target Young's modulus.

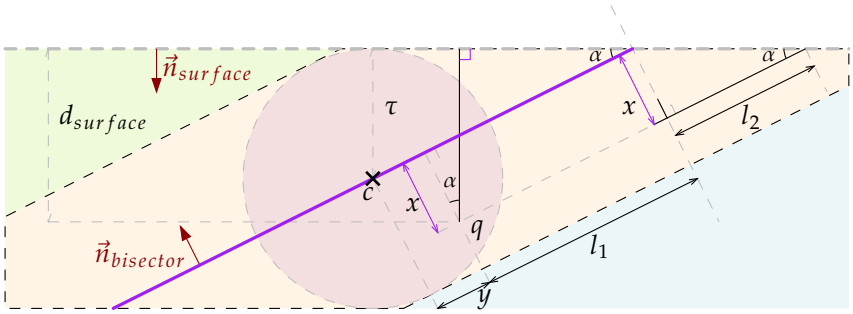
Procedural Foam with Elasticity Gradients

The process we have described in Section 5.1.3 is capable of generating graded structures, with a spatially varying Young's modulus. We exploit this property to generate objects with controlled elastic properties. The user inputs the spatially varying Young's modulus as a scalar field in space, which we denote $E(x)$, with x a point in space. Our technique does not put any requirement on how the field is encoded (e.g. implicit function or interpolated from a grid), but it is expected that the field varies smoothly compared to the size of the Voronoi cells. Violating this expectation will *not* result in an incorrect structure, but the produced elasticity gradient will not be a good match to the input field.

The target Young's modulus field is converted upon lookup into a target density following the approach proposed in Section 5.1.3 — this requires a simple tabulation computed from Figure 5.5. This directly drives ρ in Algorithm 14. Unless otherwise specified, the beam radius remains fixed at $\tau_{\min}$.

There are two main usage scenarios for our technique. A first scenario is to fill the inside of objects that have to remain rigid — simply adapting the inner density to varying stresses. In such a case the outer hull of the object remains solid and there is no additional challenge. The second scenario is to produce objects that can deform — flexible prosthesis and robot parts, toys. In such a case, printing the outer hull of the object would be detrimental to its flexibility. We therefore propose an object frame generation well suited to our approach. This frame is visible on all our 3D printed objects. It is fully procedural and only assumes that we have access to a (narrow band) distance field from the surface.

Object Frame. The key idea of our frame generation is to intersect the *faces* of the Voronoi diagram with a thick shell just below the surface, while taking care of cases where faces are almost parallel to the surface. Figure 5.11a explains our frame generation process. q is the query point and we want to know if it belongs to the solid structure or not. Here we consider two seeds s_1, s_2 among the closest seeds from q as obtained from Algorithm 14. If the point q is too close from the surface, we need to keep the intersection of the Voronoi faces with the frame (in addition to the regular Voronoi edges). Simply using the distance from q to the bisector of $\{s_1, s_2\}$ would yield the region colored in light orange, whereas we would like to select the point q iff it belongs to the beam of center c and thickness t (in light purple). However, we don't know the exact position of c, but only the distance to the border d, the distance to the bisector x, and an estimate of the angle formed by the surface and the bisector at point q, that we call α. Assuming the surface is locally planar the rest follows from basic trigonometry: knowing $l_1 + l_2 = \frac{d}{\sin(\alpha)}$, and $l_2 = \frac{x}{\tan(\alpha)}$, we can compute y and test whether $x^2 + y^2 \leqslant t^2$.



(a) Scheme of the surface query.

(b) A cylinder with 3 possible frames.

Figure 5.11 – Top: 2D planar cut orthogonal to the surface (top black line). The goal is the produce the beam which cross section is drawn as a circle. The purple line is a Voronoi face intersected by the view plane. **Bottom:** Extreme case of a frame on a small cylinder. From left to right, no frame, frame obtained by intersecting the Voronoi faces with the surface naively, frame obtained with our approach.

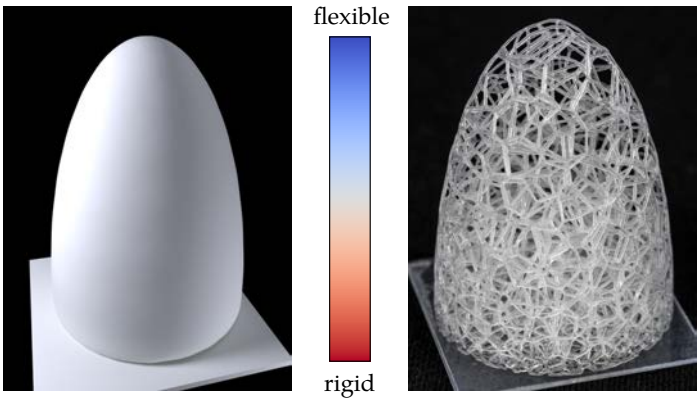


Figure 5.12 – A simple graded ellipsoid.

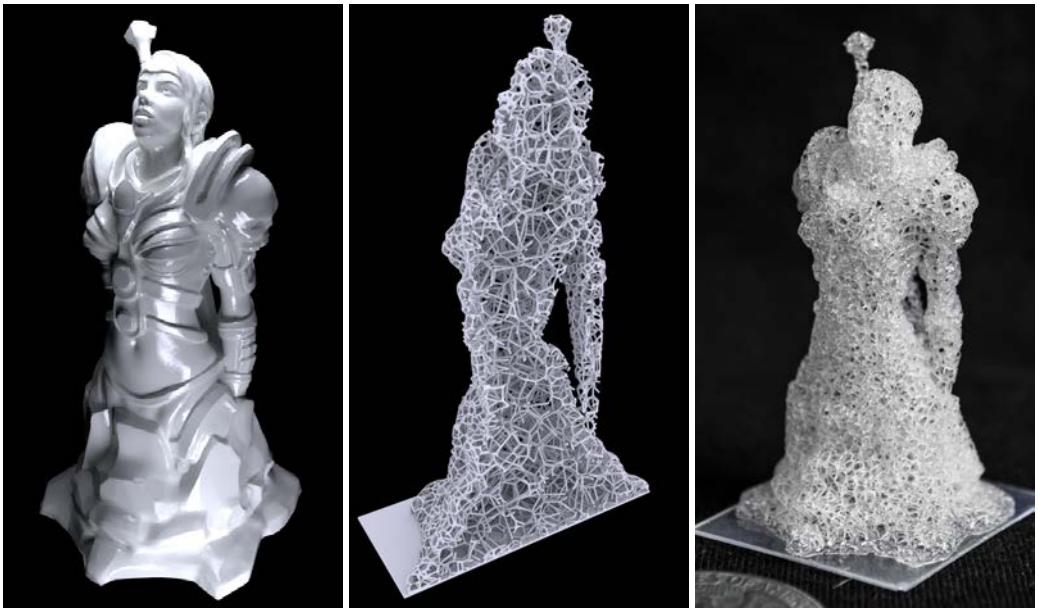


Figure 5.13 – Knight model with a dense crust, and low-density interior. Model: Knight `thing:33804` by andreas.

Figure 5.14 – Finger with articulations. Right: deformed object. Note that the inferior articulation is made wider and thus bends further. Model: `Olivier hand` (AIM@SHAPE Project).

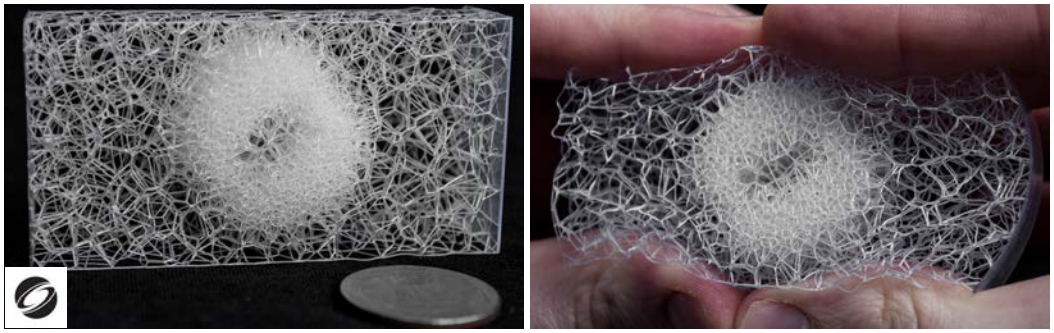


Figure 5.15 – SIGGRAPH logo driving the density field.

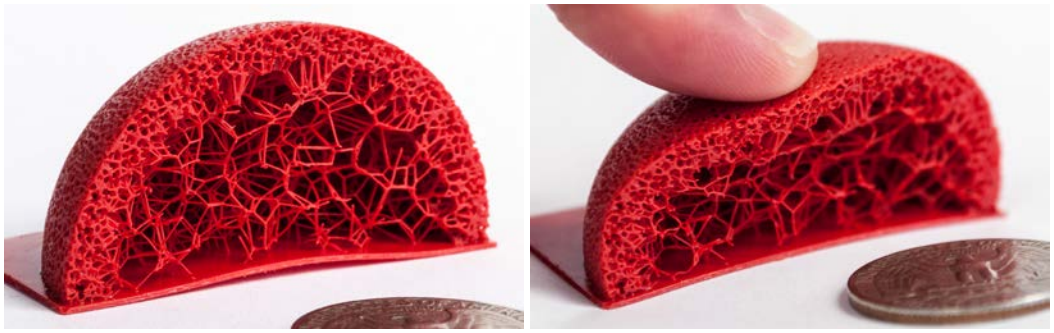


Figure 5.16 – A dome with variable density. No frame is generated on the cut revealing the interior.

Printed Results. We now apply our approach to produce a variety of 3D printed results. We use two different printers: a B9Creator with red-cherry resin and an Autodesk Ember with standard clear resin. All these prints are prepared using our image based in-house slicer using the implicit procedural foam generation.

Our SLA printers require support structures. The objects we printed do not themselves require support — a property preserved by our microstructures but for a few cases along the object frame. These cases are rare, for illustration we selected one in Figure 5.11b (top of rightmost case). Table 5.1 shows the amount of volume filtered out due to those constraints. We filter them out during out-of-core slice generation, keeping track of connected components from one slice to the next. Adapting support techniques to our microstructures is left as future work — a possibility would be to connect a standard support to the closest microstructure beam. Figure 5.20 shows the location of isolated voxels filtered out before printing on SLA printers. SLS printers do not require support and can directly print our objects.

Figure 5.12 shows a very simple case of graded material applied to a 3D model. Figure 5.13 is a detailed 3D model filled with a spatially varying structure, denser along the surface and coarser inside. Note how the frame preserves the surface details. Figure 5.16 is a similar case on a sliced sphere, revealing the deformation behavior. Figure 5.17 is a model with varying elasticity. The model deforms as

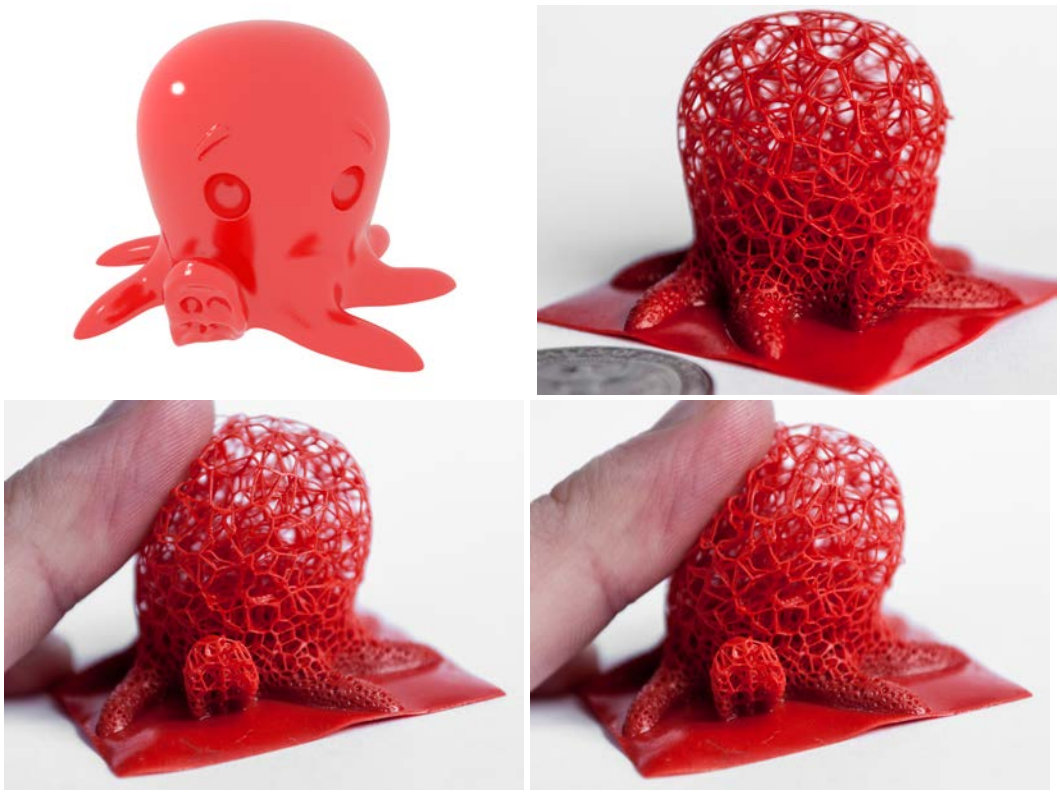


Figure 5.17 – Cute octopus. From left to right: 3D model (original), printed model, printed with deformation (finger pressure).
Model: Cute Octopus `thing:27053` by MakerBot.

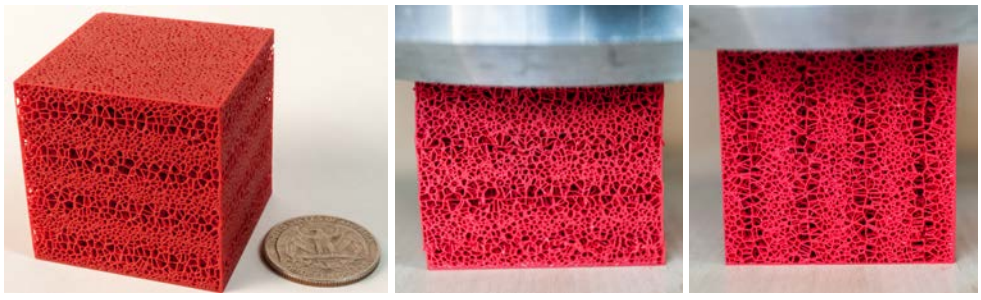


Figure 5.18 – Varying the density in one direction only can also produce parts with anisotropic behavior. The same weight of 8 kg has been applied to the cube placed in different orientations, leading to a visible deformation in the direction of the anisotropy.

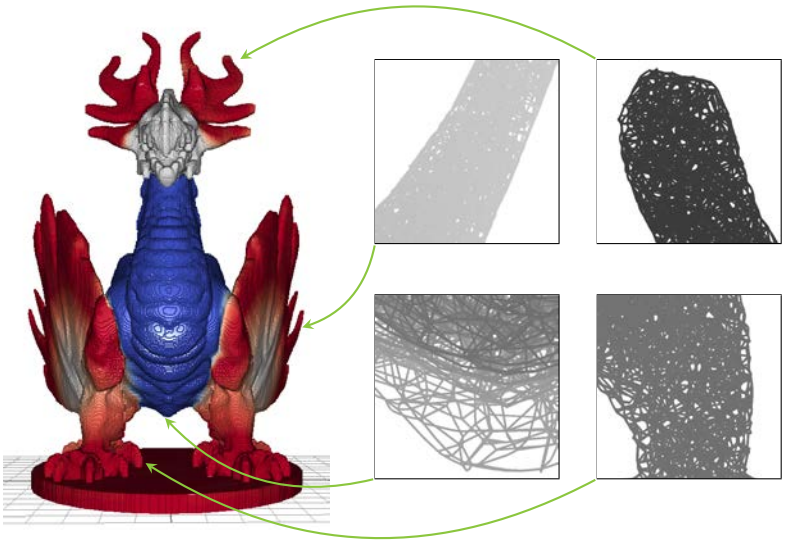


Figure 5.19 – Closeups of the depth map of a large dragon (about $1\,m^3$) with painted elasticity, sliced at a resolution of 0.05 mm. The computation domain is a volume of about 10^{12} voxels.
Model: Forest dragon (`thing:87458`) by dutchmogul.

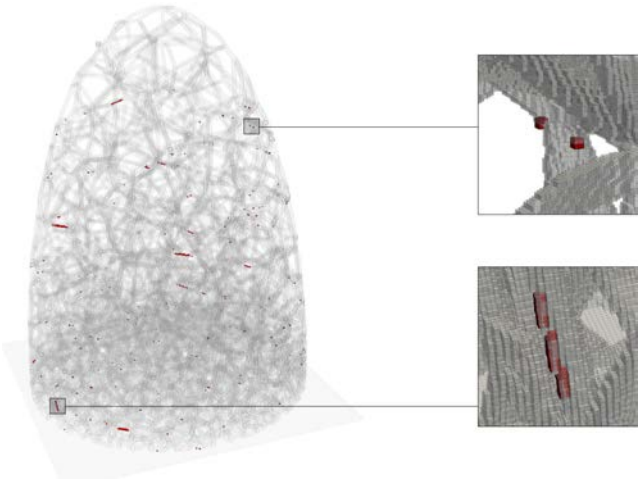


Figure 5.20 – Illustration of the isolated voxels filtered out for resin printers. Removed percentage for each example is reported in Table 5.1. Greyscaled values in voxels (section 5.1.2) have been thresholded at isovalue 0.5 to show a binary design.

Example		Extent (mm)	# Voxels	Volume	% Filtered	Time per slice (ms)
Moomin	fig. 5.1	26.7 × 40.8 × 51.9	534 × 815 × 1038	6.44 %	0.005 %	68.34
Ellipsoid	fig. 5.12	30.9 × 30.9 × 41.1	617 × 617 × 822	6.30 %	0.001 %	37.28
Knight	fig. 5.13	26.1 × 30.0 × 50.6	521 × 600 × 1011	12.50 %	0.023 %	20.25
Finger	fig. 5.14	25.0 × 23.3 × 70.5	500 × 465 × 1410	23.35 %	0.006 %	28.03
SIGGRAPH logo	fig. 5.15	20.0 × 40.0 × 80.0	400 × 800 × 1600	5.73 %	0.003 %	69.18
Half-dome	fig. 5.16	25.0 × 50.0 × 25.0	500 × 1000 × 500	19.49 %	0.025 %	71.22
Octopus	fig. 5.17	41.7 × 41.1 × 28.8	833 × 822 × 576	17.27 %	0.009 %	150.22
Anisotropic cube	fig. 5.18	40.0 × 40.0 × 40.0	800 × 800 × 800	26.86 %	0.005 %	113.52
Forest dragon	fig. 5.19	770.1 × 990.7 × 961.7	15 402 × 19 814 × 19 234	N/A	N/A	1666.91

Table 5.1 – Statistics on the examples shown in the section. Beam thickness varies between 0.2 mm and 0.4 mm.

expected when pushing its head sideways. The object frame has a limited impact on the deformation behavior, while producing a much more visually pleasing object. It is however difficult to quantify precisely the mechanical influence of the frame. Figure 5.14 illustrated an android finger with built-in flexible articulations. Figure 5.1 shows a more complex case of a painted object, where we added a support manually. Figure 5.15 shows how the elasticity field can be controlled, e.g. by the SIGGRAPH logo. Figure 5.18 illustrates how the elasticity field can be used to produce additional effects, such as anisotropic behaviors. Table 5.1 gives detailed statistics on all these prints, including average time per-slice. The throughput of our implementation averages to 7.8 Mpixel/s on the slice images on an Intel® Core™ i7-4770K @ 3.50 GHz, 16 GiB RAM with a Titan Black NVidia GPU.

Large Objects. Our approach scales trivially with object size. While we are currently limited in the size of objects we can print, we illustrate this by producing a microstructure in a model that is about one meter in size. The sliced equivalent has 5 *tera*-voxels — but of course only a single slice (15 402 × 19 814) would have to fit in the printer memory. Figure 5.19 provides closeups of the depth map obtained by raycasting the implicit structure generated by our algorithm.

Discussion and Limitations

Our technique has a number of limitations. The foams only exhibit their target properties when a sufficiently large volume is printed — a limitation shared by all approaches relying on homogenization. This is however aligned with our goal of producing dense microstructures in large objects.

Compared to regular structures the stochastic nature of the foams produces more localized stresses. We have observed that a few beams fail under large deformations, perhaps earlier than on regular structures. This is specially the case along object frames with the Ember standard resin, which is more brittle. This would require

further studies, noting that crushing behaviors of naturally occurring open-cell foams have received some interest [Gaitanaros et al. 2012].

Contrary to tile-based approaches we currently cannot provide spatially varying Poisson's ratios. Studying stochastic structures that can vary both Young's modulus and Poisson's ratio is an interesting direction of future work. Generating structures with anisotropic behaviors is another natural venue for further studies.

5.1.5 Supplemental Material

In this section, first, we present the formula of the elasticity tensor for isotropic materials. Second, we present a short background on numerical homogenization. Finally, we give the iterative algorithm to generate seeds, and describe the Python code attached in the supplemental material.

We use the following notation:

Notation	Description
E	Young's modulus
ν	Poisson's ratio
ε	Strain tensor
σ	Stress tensor
C	Elasticity tensor
$\mathbf{u}$	Displacement field
$\mathbf{f}$	Force field
$\mathbf{K}$	Stiffness matrix

Isotropic Material Tensor

$$\widehat{E} = \frac{E}{(1-2\nu)(1+\nu)} \tag{5.3}$$

$$G = \frac{E}{2(1+\nu)} \tag{5.4}$$

$$C^I(E,\nu) = \begin{pmatrix} \widehat{E}(1-\nu) & \widehat{E}\nu & \widehat{E}\nu & 0 & 0 & 0 \\ \widehat{E}\nu & \widehat{E}(1-\nu) & \widehat{E}\nu & 0 & 0 & 0 \\ \widehat{E}\nu & \widehat{E}\nu & \widehat{E}(1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & G & 0 & 0 \\ 0 & 0 & 0 & 0 & G & 0 \\ 0 & 0 & 0 & 0 & 0 & G \end{pmatrix} \tag{5.5}$$

Background on Homogenization

Homogenization is at the core of most existing work regarding microstructures [Allaire 2012]. We rely on homogenization to predict the large scale behavior of our structures. We therefore give some more precise background regarding this technique.

Homogenization efficiently determines the elasticity tensor of a periodic composite material defined from a unit tile V, having a volume $|V|$. For small deformations of an elastic material, the amount of stress σ is linearly proportional to the strain ε, as given by the elasticity tensor C:

$$\sigma = C : \varepsilon$$

The homogenized elasticity tensor C^H can be derived as [Sanchez-Palencia 1980]:

$$C^H_{rspq} = \frac{1}{|V|} \int_V C_{ijkl} \left(\varepsilon^{0(ij)}_{pq} - \varepsilon^{ij}_{pq} \right) \left(\varepsilon^{0(kl)}_{rs} - \varepsilon^{kl}_{rs} \right) \mathrm{d}V \tag{5.6}$$

$\varepsilon^{0(ij)}_{pq}$ are prescribed strain fields. $\varepsilon^{(ij)}_{pq}$ is obtained as $\varepsilon_{pq}(\mathbf{u}^{ij})$, where the displacement field $\mathbf{u}^{ij}$ is the result of solving the elasticity equations with prescribed strain.

For most problems, homogenization is performed numerically by discretization, solving the elasticity equation with the finite element method (FEM). Our method is an extension to 3D of the 2D implementation of [Andreassen and Andreasen 2014]. The elasticity equation is discretized on a regular grid with N hexahedral elements. Consider six force vectors $\mathbf{f}^{(i)}$ corresponding to prescribed unit strains (the six different strain coordinate directions in 3D). $\mathbf{u}^{0(i)}$ are the six displacement fields corresponding to unit strains, and $\mathbf{u}^{(i)}$ are the displacement fields resulting from enforcing the corresponding unit strains. The six displacement fields are obtained by solving for linear elasticity $\mathbf{K}\mathbf{u}^{(i)} = \mathbf{f}^{(i)}$, with imposed periodic boundary conditions. When the displacements have been obtained, the homogenized elasticity tensor can be found as [Andreassen and Andreasen 2014]:

$$C^H_{ij} = \frac{1}{|V|} \sum_e^N \int_{V_e} \left(\mathbf{u}^{0(i)}_e - \mathbf{u}^{(i)}_e \right)^T \mathbf{K}_e \left(\mathbf{u}^{0(j)}_e - \mathbf{u}^{(j)}_e \right) \mathrm{d}V_e \tag{5.7}$$

This tensor characterizes the linear elastic behavior of the periodic material.

Seed Generation Process

Algorithm 17 shows a stackless version, iterative method for generating seeds in a coarse grid cell.

We also provide two sample Python codes implementing the adaptive sampling method described in our work, specialized for the 2D case. The files can be found

in the `code.zip` archive accompanying this supplemental material of the original article [Martínez et al. 2016], and are organized as follows:

- The first file, `generate_seeds_recursive.py` is a 67 lines python version of our algorithm SubdivideCell, using built-in calls for generating random numbers.
- The second file, `generate_seeds_iterative.py` is a 141 lines stackless version of the same code, with explicit random number generation (using the same generator as the default random number generator of `libstdc++`). This version is amenable to a GPU implementation.

The code works with both `python2` and `python3`, and includes a plot of the result. It depends only on `numpy` and `matplotlib`.

5.1.6 Conclusion

The main advantages of our approach stem from the implicit formulation of stochastic microstructures. This has significant computational advantages by allowing for evaluation at slicing time and by avoiding to resort on global optimizations for each new object. The aperiodic and stochastic nature of the foams provide a simple and efficient way to grade the structures and to conform to target elasticity fields in space, without introducing artificial boundaries.

While we took a strict interpretation on the procedural nature of our structure generation, it is clear that other computational schemes could be envisioned. The important fundamental properties are 1) that the evaluation of the structure remains *local* and independent from the overall size of the domain and 2) that the elastic properties relate to the structure parameters through a simple relationship avoiding complex parameter fitting during evaluation.

Our approach deviates significantly from both the periodic tiling of microstructures and the optimization of macrostructures, by making a link between microstructures and procedural solid textures with controlled statistics in computer graphics. We believe there are many other such structures to be discovered, and hope our work will spark further interest in procedurally generated, stochastic microstructures.

Algorithm 17: SUBDIVIDECELLITERATIVE

Input: Starting coarse cell, density field ρ.
Output: A set of seeds, with a density driven by ρ

$N \leftarrow \emptyset$;
$d \leftarrow 0$; // depth in the quadtree/octree
$ijk \leftarrow$ coarseCell.center;
while *true* **do**
  $l \leftarrow$ coarseCell.length $\times 2^{-d}$;
  $c \leftarrow l \times ijk + l/2$; // center of the current cell
  $t \leftarrow l^3 \times \rho(c)$; // target number of seeds in cell
  if $t \leqslant 2^3$ **then**
    $I =$ RANDOMPERMUTATION(SUBCELLS(c, l));
    $n_{\min} = \lfloor t \rfloor$; // minimum number of samples to draw
    for $i \in [\![0, n_{\min}[\![$ **do**
      $N \leftarrow N \cup \{$RANDOMSAMPLEINSUBCELL$(I[i])\}$;
    $p \leftarrow$ RANDOMFLOAT$(0, 1)$;
    if $p \leqslant (t - n_{\min})$ **then**
      $N \leftarrow N \cup \{$RANDOMSAMPLEINSUBCELL$(I[n_{\min}])\}$;
    // move up the cells
    while $ijk \mod 2 = (1, 1, 1) \wedge d > 0$ **do**
      $d \leftarrow d - 1$;
      $ijk \leftarrow \lfloor ijk/2 \rfloor$;
    if $d > 0$ **then**
      // move to next cell with the same parent
      $ijk \leftarrow$ NEXTONLEVEL(ijk);
    else
      break
  else
    // go to the first child cell
    $d \leftarrow d + 1$;
    $ijk \leftarrow ijk \times 2$;
return N

5.2 Discussion and Conclusion

In the present chapter we have described an efficient approach to control the microstructural geometry of a printed model, in a manner that is suitable for additive manufacturing. The presented approach is an original application of techniques inspired from procedural texture synthesis, in an original combination that considers mechanical behavior of the synthesized structures. The use of Voronoi open cells guarantee printability of the interior details on resin-based printer, and is also amenable to other technologies such as powder-bed printers (see Figure 2.8c).

Another advantage of our technique is that it provides an efficient and easy way to grade material properties within an object. This is especially interesting for functionally graded materials, which previous methods optimizes explicitly, e.g. [Radman et al. 2013]. In contrast to standard tiling methods, our approach naturally allows the grading field to conform and adapt to the surface geometry, as exemplified in Figures 5.13 and 5.16. While our procedural approach addresses the two aforementioned challenges faced by tiling methods — cells connectivity and conforming grading —, it is not without its own shortcomings. The results we provide are only a first step, that we hope will foster further research, towards more elaborate solutions.

In the interest of evaluating the quality of present and future solutions, we feel that a immediate challenge common to every approach lies in measuring the mechanical properties of spatially varying structures. Indeed, in homogenization theory, and in most mechanical testing devices, a regular lattice with a constant density is usually expected. But how should one measure the equivalent elastic tensor of a structure with varying density printed on a complex shape like the Armadillo[1]? An orthogonal issue is to consider the effect of the skin shell on the mechanical behavior of the printed microstructures. To which extent does printing a skin with the microstructure changes its rigidity? Printing a "skin frame" versus fully solid "skin shell" surely makes the whole object stiffer, but by how much? To the best of our knowledge, it seems that this effect has been scarcely studied in the mechanical engineering literature. Some recent work that consider the influence of the skin on printed geometry are presented in [Brennan-Craddock et al. 2012; Aremu et al. 2016].

Beyond the influence on the mechanical properties of the interior microstructures, the skin at the boundary of volume can also have an adverse effect on the printing process. Indeed, while Voronoi cells that are fully enclosed in the volume are always convex, this is not the case when a cell is intersected by the surface at the boundary. This is visible on the extreme example shown Figure 5.11b. In practice, for resin printers, we are only interested in avoiding local minima in the printing direction. To this end, enforcing convex cells even near the boundary is only a sufficient condition, but not a necessary one. Note that even if all the cells are

[1] http://graphics.stanford.edu/pub/3Dscanrep/armadillo/Armadillo.ply.gz

convex, if the original input volume exhibits local minima — like the head of the Moomin in Figure 5.1 —, then no matter what partitioning scheme is used for the interior microstructures, external supports will be required to print the final shape. A key property of approach though, is that the support does not have to go inside and pollute the microstructure.

As a future work, we would like to explore different families of procedural patterns for fabrication, and relate their parameters to their physical properties. Indeed, the convex cells obtained from a Voronoi diagram are *in fine* a very restrictive class of microstructures, which does not cover a range as wide as the results featured in [Panetta et al. 2015; Schumacher et al. 2015] for example. In comparison, the procedural synthesis methods discussed in Section 2.4.1 already covers different pattern families, but the question of their equivalent elastic properties remains largely unexplored.

While only briefly mentioned in the conclusion of Section 5.1, I would like to discuss the benefits and drawbacks of using other computational schemes — like *hybrid* methods and out of core approaches —, as opposed to the purely procedural approach presented here. For example, one could imagine generating all the Voronoi seeds as a preprocessing step, and pass this 1D array to the subroutine computing the slice geometry — thereby short-circuiting the call to GatherSeeds on line 1 in Algorithm 14. This is probably acceptable for the presented type of patterns presented here, and would probably scale quite well up to, say, 10^9 seeds. However, after that limit, if you need to increase the print volume of the resolution of your model and microstructures, some extra work will be necessary to make this approach scale well. Recall that the Dragon presented in Figure 5.19 has 10^{12} voxels, and can easily handle more than 10^9 seeds. For example, one can imagine that an accelerating structure like a k-d tree will be needed to fetch the seeds around a query point in Algorithm 14. A second possibility is to simply pass the points around the current slab to the EvalStructure routine instead of the whole 1D array containing all the seeds. But we can argue that this amounts to generate seeds locally around each slice or query point, as is done by GatherSeeds in Algorithm 15. The last argument to keep in mind, is that there is an extra engineering cost to every paradigm change. To put it differently, it is easier to create a program and experiment with different microstructure families, if all the synthesis subroutines have a common interface which is a pure procedural function $\mathcal{F}(x, y, z)$.

At this point, it is interesting to make a parallel with two-scale topology optimization techniques, such as the homogenization methods mentioned Section 2.5.1. Indeed, the objective in Section 5.1 was to generate a geometry whose elastic behavior matches an input elasticity field. However, the question of how this input field is computed is left mostly untouched. For example, one could imagine taking the output of an unpenalized version of SIMP (the so-called variable thickness sheet problem), and use it as an input to our procedure described Section 5.1. In two-scale topology optimization methods, an algorithm optimizes jointly for both microstructures geometry and the macro-scale description of the design. While both parts

are optimized together, they are still decoupled: without extra constraints, nothing ensures that microstructures in neighboring cells have a compatible connectivity. There are however recent topology optimization techniques, such as [Alexandersen and Lazarov 2015], that seek to optimize manufacturable microstructural details directly without separating the micro- and macro-scale behavior via homogenization. Note also that even with a few unit cells, Panetta et al. [2015] reported a good agreement between predicted homogenized material properties and measured behavior of printed samples.

As the size of printed models increases, one can also ponder the effects of the limited print resolutions on fabricated models. As explained in the introduction of [Cook et al. 2007a], one can distinguish 3 sources of errors when computing the solution of a mathematical model representing a physical phenomenon: *modeling error*, *discretization error*, and *numerical error*. Modeling errors represent the approximation made by the mathematical model itself, usually the PDE describing the mechanical behavior in the continuous domain. It can be alleviated by improving the model, e.g. linear material vs nonlinear materials. Discretization errors are introduced when the continuous description of the problem is discretized in several elements, which yield a piecewise linear or polynomial representations. It can be improved by increasing the number of elements, or the order of their basis function. Numerical errors are intrinsically bound to the limited precision of arithmetical operations carried out on the computer. They can be alleviated by using more numerically robust routines, especially when solving linear systems with a high condition number. The use of exact arbitrary-precision arithmetic is often impractical for solving large finite element problems.

In this context, the fabrication process introduces a fourth type of error, that could be called *machining error*. It is intrinsic to the inaccuracies of the fabrication processes: geometrical defects or mechanical play in the frame or carriage of a filament; calibration precision on resin and DLP printers, etc. This results in inaccuracies in the printed shapes, with the consequence that no two prints can be exactly the same. The extent of these inaccuracies is probably relatively small, hardly visible, and difficult to measure with limited precision equipment. However, for industrial applications, it might become a concern. To this end, a few recent works are striving to directly model the physical behavior of filament deposition printers, via finite element analysis [Guessasma et al. 2015; Liu and Shapiro 2016]. Modeling uncertainties in the manufacturing process is also an important part of robust structural optimization methods, see e.g. surveys [Beyer and Sendhoff 2007; Schuëller and Jensen 2008; Wang et al. 2011], and more recently [Lazarov et al. 2012a,b; Zhou et al. 2014a]. Note that there, the *machining error* really encompasses the inaccuracies of the physical process — like the resin curing, or the filament deposition. The finite resolution of the images representing a slice (with resin printers), or the piecewise linear nature of toolpaths in FDM printers, should be considered a part of the *discretization error* rather than the *machining error*.

Chapter 6

Conclusion

In this thesis several methods for controllable shape synthesis were presented. Chapter 3 describes algorithms for treating complex shapes before fabrication. In particular, we introduced an original method for generating support structures that are well suited for filament printers, based on the idea of bridges. Our approach provides increased stability and robustness compared to other sparse support methods, for a similar material waste and print time.

In Chapter 4, we introduced methods for by-example shape synthesis that account for both appearance and mechanical behavior of the generated structures. This opens up new directions of research, where by-example synthesis methods not only have to improve the appearance of a model, but need to be combined with shape optimization techniques in a meaningful manner. We have presented different synthesis algorithms, based on pixel neighborhoods on a voxelized surface in Section 4.1, joint topology and appearance optimization in Section 4.2, and aggregate element synthesis in Section 4.3.

In Chapter 5, we proposed a technique for controlling the elasticity of a structure by efficiently generating printable microstructures through a procedural function. The proposed solution scales up well with the size of a model, and is amenable to online synthesis, as the microstructures can be streamed directly to the printers.

Impact. Since the first publication of our bridge support structures in [Dumas et al. 2014], the article has been cited more than 20 times in 2 years. Unfortunately, there is no available open-source implementation of our support synthesis algorithm, which could have helped a wider adoption of our method from the additive manufacturing community. While the algorithm itself not exceedingly complex, its implementation needs to be done carefully. Our reference implementation in the original publication is tightly integrated our slicing software *IceSL*, especially for the computation of the support points and the post-synthesis boolean operations with the original model. This makes it difficult to provide an independent open-source code directly exploitable by the community. Furthermore, the current implementation within *IceSL* could be improved, in particular the part that precomputes the acceleration structure for collision detection. In its current state, the code would need further engineering work before it can be put into production into a public release of *IceSL*.

Figure 6.1 – A bottle and a vase generated with our technique from Section 4.1. Models made for the company *Creative Industries* [link].

Regarding Chapter 4, it is worth mentioning that the technique from [Dumas et al. 2015] has already stirred interest from a local company called *Creative Industries*. Two examples of fabricated objects are shown in Figure 6.1. The patterns used for these two objects demonstrate the interest for discrete elements as a design tool (e.g. the bubbles in the bottle, and the arrows in the vase). As our work in Chapter 4 was published more recently, it is also more difficult to judge its impact. However, as discussed in Section 4.4, there has been a number of concurrent or subsequent works on closely related problems. Whether these works have been inspired by ours, or just happened to study similar problems, is irrelevant. The point is that it shows the increasing interest for appearance control in shape synthesis for fabrication.

Similarly, it is still early to judge the impact of our work in Chapter 5, but given the expanding capabilities of 3D printing technologies, we can only hope that the controlled procedural approach presented in [Martínez et al. 2016] will inspire exciting applications in the future.

Perspectives and Future Work. More specialized discussions have already been provided in individual chapters, so the following contains only general comments about the work presented in this manuscript. In general, the techniques we have developed are focused on the automatic optimization of a certain problem. However, there are cases where interactive input from the user is actually desired. For example, to guide the support generation to avoid certain sensitive features of a model. For by-example synthesis, having interactive feedback when drawing constraints for a shape can help the design process. Indeed, works such as [Zehnder et al. 2016] have

focused on interactive design of curve network suited for fabrication. The same goes for our microstructure synthesis algorithm, which is fully automated, although the input field can be painted by the user in an interactive application. Other approaches, such as the Element[1] software by *nTopology*, provide an interactive lattice editor for generating microstructures in a model. Recent works in metamaterial design, such as [Ion et al. 2016; Vidimce et al. 2016], attest the interest in fabrication-oriented design software for microstructures.

In conclusion, I believe that automatic methods for the synthesis of optimal structures are essential to help designing shapes for fabrication. However, current topology optimization techniques are not yet amenable to interactive synthesis of detailed structures in 3D, despite recent efforts in this direction [Aage et al. 2013; Nobel-Jørgensen et al. 2015]. Providing extra control over the appearance of the produced results adds another set of challenges, and I hope that in the future, more interactive applications for large-scale controllable shape optimization in 3D will emerge.

[1] http://www.ntopology.com/

Bibliography

Aage, N.; Andreassen, E. and Lazarov, B. S. (2015). "Topology Optimization Using PETSc: An Easy-To-Use, Fully Parallel, Open Source Topology Optimization Framework". *Structural and Multidisciplinary Optimization* 51.3, 565–572. doi: 10.1007/s00158-014-1157-0.

Aage, N.; Nobel-Jørgensen, M.; Andreasen, C. S. and Sigmund, O. (2013). "Interactive Topology Optimization on Hand-Held Devices". *Structural and Multidisciplinary Optimization* 47.1, 1–6. doi: 10.1007/s00158-012-0827-z.

Alexander, P.; Allen, S. and Dutta, D. (1998). "Part Orientation and Build Cost Determination in Layered Manufacturing". *Computer-Aided Design* 30.5, 343–356.

Alexandersen, J. and Lazarov, B. S. (2015). "Topology Optimisation of Manufacturable Microstructural Details Without Length Scale Separation Using a Spectral Coarse Basis Preconditioner". *Computer Methods in Applied Mechanics and Engineering* 290, 156–182. doi: 10.1016/j.cma.2015.02.028.

Allaire, G.; Jouve, F. and Michailidis, G. (2016). "Thickness Control in Structural Optimization via a Level Set Method". *Structural and Multidisciplinary Optimization* 53.6, 1349–1382. doi: 10.1007/s00158-016-1453-y.

Allaire, G. and Pantz, O. (2006). "Structural Optimization With FreeFem++". *Structural and Multidisciplinary Optimization* 32.3, 173–181. doi: 10.1007/s00158-006-0017-y.

Allaire, G. (2007). *Conception Optimale De Structures*. Vol. 58. Springer Berlin Heidelberg. doi: 10.1007/978-3-540-36856-4.

Allaire, G. (2012). *Shape Optimization by the Homogenization Method*. Vol. 146. Springer Science & Business Media.

Allaire, G.; De Gournay, F.; Jouve, F. and Toader, A. (2005). "Structural Optimization Using Topological and Shape Sensitivity via a Level Set Method". *Control and cybernetics* 34.1, 59. eprint: http://oxygene.ibspan.waw.pl:3000/contents/export?filename=2005-1-03_allaire_et_al.pdf.

Allaire, G.; Jouve, F. and Toader, A. (2002). "A Level-Set Method for Shape Optimization". *Comptes Rendus Mathematique* 334.12, 1125–1130. doi: 10.1016/s1631-073x(02)02412-3.

Allaire, G.; Jouve, F. and Toader, A. (2004). "Structural Optimization Using Sensitivity Analysis and a Level-Set Method". *Journal of Computational Physics* 194.1, 363–393. doi: 10.1016/j.jcp.2003.09.032.

Allen, S. and Dutta, D. (1995). "Determination and Evaluation of Support Structures in Layered Manufacturing."

Allison, J. W.; Chen, T. P.; Cohen, A. L.; Smalley, D. R.; Snead, D. E. and Vorgitch, T. J. (1988). *Boolean Layer Comparison Slice*. US Patent 5854748, 3D Systems Inc.

AlMeraj, Z.; Kaplan, C. S. and Asente, P. (2013a). "Patch-Based Geometric Texture Synthesis". *Proceedings of the Symposium on Computational Aesthetics - CAE '13*. Association for Computing Machinery (ACM). doi: 10.1145/2487276.2487278.

AlMeraj, Z.; Kaplan, C. S. and Asente, P. (2013b). "Towards Effective Evaluation of Geometric Texture Synthesis Algorithms". *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering - NPAR '13*. Association for Computing Machinery (ACM). doi: 10.1145/2486042.2486043.

Amir, O.; Aage, N. and Lazarov, B. S. (2014). "On Multigrid-CG for Efficient Topology Optimization". *Structural and Multidisciplinary Optimization* 49.5, 815–829. doi: 10.1007/s00158-013-1015-5.

Ananiev, S. (2005). "On Equivalence Between Optimality Criteria and Projected Gradient Methods With Application to Topology Optimization Problem". *Multibody System Dynamics* 13.1, 25–38. doi: 10.1007/s11044-005-2530-y.

André, J.; Le Mehaute, A. and De Witte, O. (1984). "Dispositif Pour Réaliser Un Modèle De Pièce Industrielle". 411. url: http://bases-brevets.inpi.fr/fr/document/FR2567668/publications.html.

Andreassen, E. and Andreasen, C. S. (2014). "How to Determine Composite Material Properties Using Numerical Homogenization". *Computational Materials Science* 83, 488–495. doi: 10.1016/j.commatsci.2013.09.006.

Andreassen, E.; Clausen, A.; Schevenels, M.; Lazarov, B. S. and Sigmund, O. (2011). "Efficient Topology Optimization in MATLAB Using 88 Lines of Code". *Structural and Multidisciplinary Optimization* 43.1, 1–16. doi: 10.1007/s00158-010-0594-7.

Andreassen, E.; Lazarov, B. S. and Sigmund, O. (2014). "Design of Manufacturable 3D Extremal Elastic Microstructure". *Mechanics of Materials* 69.1, 1–10. doi: 10.1016/j.mechmat.2013.09.018.

Aremu, A. O.; Maskery, I. A.; Tuck, C. J.; Ashcroft, I. A.; Wildman, R. D. and Hague, R. J. M. (2016). "Effects of Net and Solid Skins on Self-Supporting Lattice Structures". *Challenges in Mechanics of Time Dependent Materials, Volume 2*. Springer Science + Business Media, 83–89. doi: 10.1007/978-3-319-22443-5_10.

Arora, J. and Wang, Q. (2005). "Review of Formulations for Structural and Mechanical System Optimization". *Structural and Multidisciplinary Optimization* 30.4, 251–272. doi: 10.1007/s00158-004-0509-6.

Ashikhmin, M. (2001). "Synthesizing Natural Textures". *Proceedings of the 2001 symposium on Interactive 3D graphics - SI3D '01*. Association for Computing Machinery (ACM). doi: 10.1145/364338.364405.

Attene, M. (2015). "Shapes in a Box: Disassembling 3D Objects for Efficient Packing and Fabrication". *Computer Graphics Forum* 34.8, 64–76. doi: 10.1111/cgf.12608.

Autodesk (2016). *Within*. www.autodesk.com/products/within.

Bächer, M.; Bickel, B.; James, D. L. and Pfister, H. (2012). "Fabricating Articulated Characters From Skinned Meshes". *ACM Transactions on Graphics* 31.4, 1–9. doi: 10.1145/2185520.2185543.

Bächer, M.; Whiting, E.; Bickel, B. and Sorkine-Hornung, O. (2014). "Spin-It: Optimizing Moment of Inertia for Spinnable Objects". *ACM Transactions on Graphics* 33.4, 1–10. doi: 10.1145/2601097.2601157.

Barla, P.; Breslav, S.; Thollot, J.; Sillion, F. and Markosian, L. (2006). "Stroke Pattern Analysis and Synthesis". *Computer Graphics Forum*. Vol. 25. 3. Wiley Online Library, 663–671. eprint: https://hal.inria.fr/inria-00362883/document.

Barnes, C.; Shechtman, E.; Finkelstein, A. and Goldman, D. B. (2009). "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing". *ACM SIGGRAPH 2009 papers on - SIGGRAPH '09*. Association for Computing Machinery (ACM). doi: 10.1145/1576246.1531330.

Bendsøe, M. P. (1989). "Optimal Shape Design as a Material Distribution Problem". *Structural Optimization* 1.4, 193–202. doi: 10.1007/bf01650949.

Bendsøe, M. P. and Sigmund, O. (1999). "Material Interpolation Schemes in Topology Optimization". *Archive of Applied Mechanics (Ingenieur Archiv)* 69.9-10, 635–654. doi: 10.1007/s004190050248.

Bendsøe, M. P. and Kikuchi, N. (1988). "Generating Optimal Topologies in Structural Design Using a Homogenization Method". *Computer Methods in Applied Mechanics and Engineering* 71.2, 197–224. doi: 10.1016/0045-7825(88)90086-2.

Bendsøe, M. P. and Sigmund, O. (2004). *Topology Optimization: Theory, Methods and Applications*. Springer Science + Business Media. doi: 10.1007/978-3-662-05086-6.

Beyer, H. and Sendhoff, B. (2007). "Robust Optimization – a Comprehensive Survey". *Computer Methods in Applied Mechanics and Engineering* 196.33-34, 3190–3218. doi: 10.1016/j.cma.2007.03.003.

Bharaj, G.; Levin, D. I. W.; Tompkin, J.; Fei, Y.; Pfister, H.; Matusik, W. and Zheng, C. (2015). "Computational Design of Metallophone Contact Sounds". *ACM Transactions on Graphics* 34.6, 1–13. doi: 10.1145/2816795.2818108.

Bhat, P.; Ingram, S. and Turk, G. (2004). "Geometric Texture Synthesis by Example". *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing - SGP '04*. Association for Computing Machinery (ACM). doi: 10.1145/1057432.1057437.

Bickel, B.; Bächer, M.; Otaduy, M. A.; Lee, H. R.; Pfister, H.; Gross, M. and Matusik, W. (2010). "Design and Fabrication of Materials With Desired Deformation Behavior". *ACM Transactions on Graphics* 29.4, 1. doi: 10.1145/1778765.1778800.

Bostanabad, R.; Bui, A. T.; Xie, W.; Apley, D. W. and Chen, W. (2016). "Stochastic Microstructure Characterization and Reconstruction via Supervised Learning". *Acta Materialia* 103, 89–102. doi: 10.1016/j.actamat.2015.09.044.

Bourdin, B. (2001). "Filters in Topology Optimization". *International Journal for Numerical Methods in Engineering* 50.9, 2143–2158. doi: 10.1002/nme.116.

Bowers, J.; Wang, R.; Wei, L. and Maletz, D. (2010). "Parallel Poisson Disk Sampling With Spectrum Analysis on Surfaces". *ACM Trans. Graph.* 29.6, 166:1–166:10. doi: 10.1145/1882261.1866188.

Brackett, D. J.; Ashcroft, I. A.; Wildman, R. D. and Hague, R. J. M. (2014). "An Error Diffusion Based Method to Generate Functionally Graded Cellular

Structures". *Computers & Structures* 138, 102–111. DOI: 10.1016/j.compstruc.2014.03.004.

Brackett, D.; Ashcroft, I. and Hague, R. (2011). "Topology Optimization for Additive Manufacturing". *Proceedings of the Solid Freeform Fabrication Symposium, Austin, TX*, 348–362. eprint: http://sffsymposium.engr.utexas.edu/Manuscripts/2011/2011-27-Brackett.pdf.

Brennan-Craddock, J.; Brackett, D.; Wildman, R. and Hague, R. (2012). "The Design of Impact Absorbing Structures for Additive Manufacture". *Journal of Physics: Conference Series* 382, 012042. DOI: 10.1088/1742-6596/382/1/012042.

Brennan-Craddock, J. (2011). "The Investigation of a Method to Generate Conformal Lattice Structures for Additive Manufacturing". Doctoral dissertation. © James Brennan-Craddock.

Bruggi, M. and Duysinx, P. (2012). "Topology Optimization for Minimum Weight With Compliance and Stress Constraints". *Structural and Multidisciplinary Optimization* 46.3, 369–384. DOI: 10.1007/s00158-012-0759-7.

Bruns, T. E. and Tortorelli, D. A. (2001). "Topology Optimization of Non-Linear Elastic Structures and Compliant Mechanisms". *Computer Methods in Applied Mechanics and Engineering* 190.26-27, 3443–3459. DOI: 10.1016/s0045-7825(00)00278-4.

Brunton, A.; Arikan, C. A. and Urban, P. (2015). "Pushing the Limits of 3D Color Printing: Error Diffusion With Translucent Materials". *ACM Transactions on Graphics* 35.1, 1–13. DOI: 10.1145/2832905.

Bruyneel, M. and Duysinx, P. (2005). "Note on Topology Optimization of Continuum Structures Including Self-Weight". *Structural and Multidisciplinary Optimization* 29.4, 245–256. DOI: 10.1007/s00158-004-0484-y.

Bruyneel, M.; Duysinx, P. and Fleury, C. (2002). "A Family of MMA Approximations for Structural Optimization". *Structural and Multidisciplinary Optimization* 24.4, 263–276. DOI: 10.1007/s00158-002-0238-7.

Burger, M.; Hackl, B. and Ring, W. (2004). "Incorporating Topological Derivatives Into Level Set Methods". *Journal of Computational Physics* 194.1, 344–362. DOI: 10.1016/j.jcp.2003.09.033.

Burger, M. and Osher, S. J. (2005). "A Survey on Level Set Methods for Inverse Problems and Optimal Design". *European Journal of Applied Mathematics* 16.2, 263–301. DOI: 10.1017/s0956792505006182.

Busto, P. P.; Eisenacher, C.; Lefebvre, S. and Stamminger, M. (2010). "Instant Texture Synthesis by Numbers". *VMV*, 81–85. eprint: http://alice.loria.fr/publications/papers/2010/TEXNBR/InstantTBN.pdf.

Cadman, J. E.; Zhou, S.; Chen, Y. and Li, Q. (2013). "On Design of Multi-Functional Microstructural Materials". *Journal of Materials Science* 48.1, 51–66. DOI: 10.1007/s10853-012-6643-4.

Calì, J.; Calian, D. A.; Amati, C.; Kleinberger, R.; Steed, A.; Kautz, J. and Weyrich, T. (2012). "3D-Printing of Non-Assembly, Articulated Models". *ACM Transactions on Graphics* 31.6, 1. DOI: 10.1145/2366145.2366149.

Carpenter, L. (1984). "The A-Buffer, an Antialiased Hidden Surface Method". *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*

- *SIGGRAPH '84*. Association for Computing Machinery (ACM). DOI: 10.1145/800031.808585.

Ceylan, D.; Li, W.; Mitra, N. J.; Agrawala, M. and Pauly, M. (2013). "Designing and Fabricating Mechanical Automata From Mocap Sequences". *ACM Transactions on Graphics* 32.6, 1–11. DOI: 10.1145/2508363.2508400.

Chalasani, K.; Jones, L. and Roscoe, L. (1995). "Support Generation for Fused Deposition Modeling". *Solid Freeform Fabrication Symposium*, 229–241.

Challis, V. J. (2010). "A Discrete Level-Set Topology Optimization Code Written in Matlab". *Structural and Multidisciplinary Optimization* 41.3, 453–464. DOI: 10.1007/s00158-009-0430-0.

Challis, V. J.; Roberts, A. P. and Grotowski, J. F. (2014). "High Resolution Topology Optimization Using Graphics Processing Units (GPUs)". *Structural and Multidisciplinary Optimization* 49.2, 315–325. DOI: 10.1007/s00158-013-0980-z.

Chen, D.; Levin, D. I. W.; Didyk, P.; Sitthi-Amorn, P. and Matusik, W. (2013). "Spec2Fab: A Reducer-Tuner Model for Translating Specifications to 3D Prints". *ACM Transactions on Graphics* 32.4, 1. DOI: 10.1145/2461912.2461994.

Chen, D.; Levin, D. I. W.; Sueda, S. and Matusik, W. (2015a). "Data-Driven Finite Elements for Geometry and Material Design". *ACM Transactions on Graphics* 34.4, 74:1–74:10. DOI: 10.1145/2766889.

Chen, J.; Freytag, M. and Shapiro, V. (2008a). "Shape Sensitivity of Constructively Represented Geometric Models". *Computer Aided Geometric Design* 25.7, 470–488. DOI: 10.1016/j.cagd.2008.01.005.

Chen, J.; Shapiro, V.; Suresh, K. and Tsukanov, I. (2007). "Shape Optimization With Topological Changes and Parametric Control". *International Journal for Numerical Methods in Engineering* 71.3, 313–346. DOI: 10.1002/nme.1943.

Chen, J. and Wang, B. (2010). "High Quality Solid Texture Synthesis Using Position and Index Histogram Matching". *The Visual Computer* 26.4, 253–262. DOI: 10.1007/s00371-009-0408-3.

Chen, W.; Zhang, X.; Xin, S.; Xia, Y.; Lefebvre, S. and Wang, W. (2016). "Synthesis of Filigrees for Digital Fabrication". *ACM Transactions on Graphics (Proc. SIGGRAPH)* 35.4.

Chen, X.; Zheng, C.; Xu, W. and Zhou, K. (2014). "An Asymptotic Numerical Method for Inverse Elastic Shape Design". *ACM Transactions on Graphics* 33.4, 1–11. DOI: 10.1145/2601097.2601189.

Chen, X.; Zhang, H.; Lin, J.; Hu, R.; Lu, L.; Huang, Q.; Benes, B.; Cohen-Or, D. and Chen, B. (2015b). "Dapper: Decompose-And-Pack for 3D Printing". *ACM Transactions on Graphics* 34.6, 1–12. DOI: 10.1145/2816795.2818087.

Chen, Y.; Davis, T. A.; Hager, W. W. and Rajamanickam, S. (2008b). "Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate". *ACM Trans. Math. Softw.* 35.3, 22:1–22:14. DOI: 10.1145/1391989.1391995.

Chen, Y. (2007). "3D Texture Mapping for Rapid Manufacturing". *Computer-Aided Design and Applications* 4.6, 761–771. DOI: 10.1080/16864360.2007.10738509.

Chen, Y. and Wang, C. C. L. (2011). "Uniform Offsetting of Polygonal Model Based on Layered Depth-Normal Images". *Computer-Aided Design* 43.1, 31–46. DOI: 10.1016/j.cad.2010.09.002.

Cheng, G.; Mei, Y. and Wang, X. (2006). "A Feature-Based Structural Topology Optimization Method". *IUTAM Symposium on Topological Design Optimization of Structures, Machines and Materials*. Springer, 505–514.

Cheng, W.; Fuh, J. Y. H.; Nee, A. Y. C.; Wong, Y. S.; Loh, H. T. and Miyazawa, T. (1995). "Multi-Objective Optimization of Part - Building Orientation in Stereolithography". *Rapid Prototyping Journal* 1, 12–23.

Christiansen, A. N.; Bærentzen, J. A.; Nobel-Jørgensen, M.; Aage, N. and Sigmund, O. (2015a). "Combined Shape and Topology Optimization of 3D Structures". *Computers & Graphics* 46, 25–35. doi: 10.1016/j.cag.2014.09.021.

Christiansen, A. N.; Nobel-Jørgensen, M.; Aage, N.; Sigmund, O. and Bærentzen, J. A. (2014). "Topology Optimization Using an Explicit Interface Representation". *Structural and Multidisciplinary Optimization* 49.3, 387–399. doi: 10.1007/s00158-013-0983-9.

Christiansen, A. N.; Schmidt, R. and Bærentzen, J. A. (2015b). "Automatic Balancing of 3D Models". *Computer-Aided Design* 58, 236–241. doi: 10.1016/j.cad.2014.07.009.

Cignoni, P.; Gobbetti, E.; Pintus, R. and Scopigno, R. (2008). "Color Enhancement for Rapid Prototyping". *VAST*, 9–16. eprint: https://pdfs.semanticscholar.org/0d54/b1fc743aad3045b01f5177b40cf944fa641e.pdf.

Cignoni, P.; Pietroni, N.; Malomo, L. and Scopigno, R. (2014). "Field-Aligned Mesh Joinery". *ACM Transactions on Graphics* 33.1, 1–12. doi: 10.1145/2537852.

Clausen, A.; Aage, N. and Sigmund, O. (2014). "Topology Optimization With Flexible Void Area". *Structural and Multidisciplinary Optimization* 50.6, 927–943. doi: 10.1007/s00158-014-1109-8.

Clausen, A.; Aage, N. and Sigmund, O. (2015). "Topology Optimization of Coated Structures and Material Interface Problems". *Computer Methods in Applied Mechanics and Engineering* 290, 524–541. doi: 10.1016/j.cma.2015.02.011.

Clausen, A.; Aage, N. and Sigmund, O. (2016). "Exploiting Additive Manufacturing Infill in Topology Optimization for Improved Buckling Load". *Engineering* 2.2, 250–257. eprint: http://engineering.org.cn/EN/article/downloadArticleFile.do?attachType=PDF\&id=12285.

Conn, A. R.; Gould, N. I. M. and Toint, P. (1991). "A Globally Convergent Augmented Lagrangian Algorithm for Optimization With General Constraints and Simple Bounds". *SIAM Journal on Numerical Analysis* 28.2, 545–572. doi: 10.1137/0728030.

Cook, R. D.; Malkus, D. S.; Plesha, M. E. and Witt, R. J. (2007a). *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons.

Cook, R. L. (1986). "Stochastic Sampling in Computer Graphics". *ACM Transactions on Graphics* 5.1, 51–72. doi: 10.1145/7529.8927.

Cook, R. L.; Halstead, J.; Planck, M. and Ryu, D. (2007b). "Stochastic Simplification of Aggregate Detail". *ACM Transactions on Graphics* 26.3, 79. doi: 10.1145/1276377.1276476.

Coros, S.; Thomaszewski, B.; Noris, G.; Sueda, S.; Forberg, M.; Sumner, R. W.; Matusik, W. and Bickel, B. (2013). "Computational Design of Mechanical Characters". *ACM Transactions on Graphics* 32.4, 1. doi: 10.1145/2461912.2461953.

Dai, D.; Riemenschneider, H. and Van Gool, L. (2014). "The Synthesizability of Texture Examples". *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Institute of Electrical & Electronics Engineers (IEEE). DOI: 10.1109/cvpr.2014.387.

Dapogny, C. (2013). "Shape Optimization, Level Set Methods on Unstructured Meshes and Mesh Evolution". Doctoral dissertation. Université Pierre et Marie Curie-Paris 6.

Dapogny, C.; Faure, A.; Michailidis, G.; Allaire, G.; Couvelas, A. and Estevez, R. (2016). "Geometric Constraints for Shape and Topology Optimization in Architectural Design". working paper or preprint.

Deaton, J. D. and Grandhi, R. V. (2014). "A Survey of Structural and Multidisciplinary Continuum Topology Optimization: Post 2000". *Structural and Multidisciplinary Optimization* 49.1, 1–38. DOI: 10.1007/s00158-013-0956-z.

Delfs, P.; Töws, M. and Schmid, H.
bibinitperiod (2016). "Optimized Build Orientation of Additive Manufactured Parts for Improved Surface Quality and Build Time". *Additive Manufacturing*. DOI: 10.1016/j.addma.2016.06.003.

Deng, J. and Chen, W. (2016). "Design for Structural Flexibility Using Connected Morphable Components Based Topology Optimization". *Science China Technological Sciences* 59.6, 839–851. DOI: 10.1007/s11431-016-6027-0.

Derouet-Jourdan, A.; Bertails-Descoubes, F.; Daviet, G. and Thollot, J. (2013). "Inverse Dynamic Hair Modeling With Frictional Contact". *ACM Transactions on Graphics* 32.6, 1–10. DOI: 10.1145/2508363.2508398.

Derouet-Jourdan, A.; Bertails-Descoubes, F. and Thollot, J. (2010). "Stable Inverse Dynamic Curves". *ACM Transactions on Graphics* 29.6, 1. DOI: 10.1145/1882261.1866159.

Deuss, M.; Panozzo, D.; Whiting, E.; Liu, Y.; Block, P.; Sorkine-Hornung, O. and Pauly, M. (2014). "Assembling Self-Supporting Structures". *ACM Transactions on Graphics* 33.6, 1–10. DOI: 10.1145/2661229.2661266.

Díaz, A. and Sigmund, O. (1995). "Checkerboard Patterns in Layout Optimization". *Structural Optimization* 10.1, 40–45. DOI: 10.1007/bf01743693.

Dijk, N. P. van; Maute, K.; Langelaar, M. and Keulen, F. van (2013). "Level-Set Methods for Structural Topology Optimization: A Review". *Structural and Multidisciplinary Optimization* 48.3, 437–472. DOI: 10.1007/s00158-013-0912-y.

Dinh, H. Q.; Gelman, F.; Lefebvre, S. and Claux, F. (2015). "Modeling and Toolpath Generation for Consumer-Level 3D Printing". *ACM SIGGRAPH 2015 Courses on - SIGGRAPH '15*. Association for Computing Machinery (ACM). DOI: 10.1145/2776880.2792702.

Dischler, J.; Maritaud, K.; Lévy, B. and Ghazanfarpour, D. (2002). "Texture Particles". *Computer Graphics Forum* 21.3, 401–410. DOI: 10.1111/1467-8659.t01-1-00600.

Dong, Y.; Lefebvre, S.; Tong, X. and Drettakis, G. (2008). "Lazy Solid Texture Synthesis". *Computer Graphics Forum* 27.4, 1165–1174. DOI: 10.1111/j.1467-8659.2008.01254.x.

Dong, Y.; Wang, J.; Pellacini, F.; Tong, X. and Guo, B. (2010). "Fabricating Spatially-Varying Subsurface Scattering". *ACM Transactions on Graphics* 29.4, 1. doi: 10.1145/1778765.1778799.

Du, S.; Hu, S. and Martin, R. R. (2013). "Semiregular Solid Texturing From 2D Image Exemplars". *IEEE Transactions on Visualization and Computer Graphics* 19.3, 460–469. doi: 10.1109/tvcg.2012.129.

Duarte, L. S.; Celes, W.; Pereira, A.; Menezes, I. F. M. and Paulino, G. H. (2015). "PolyTop++: An Efficient Alternative for Serial and Parallel Topology Optimization on CPUs & GPUs". *Structural and Multidisciplinary Optimization* 52.5, 845–859. doi: 10.1007/s00158-015-1252-x.

Dumas, J.; Hergel, J. and Lefebvre, S. (2014). "Bridging the Gap: Automated Steady Scaffoldings for 3D Printing". *ACM Trans. Graph.* 33.4, 98:1–98:10. doi: 10.1145/2601097.2601153.

Dumas, J.; Lu, A.; Lefebvre, S.; Wu, J. and Dick, C. (2015). "By-Example Synthesis of Structurally Sound Patterns". *ACM Trans. Graph.* 34.4, 137:1–137:12. doi: 10.1145/2766984.

Duysinx, P. and Bendsøe, M. P. (1998). "Topology Optimization of Continuum Structures With Local Stress Constraints". *International Journal for Numerical Methods in Engineering* 43.8, 1453–1478. doi: 10.1002/(sici)1097-0207(19981230)43:8<1453::aid-nme480>3.0.co;2-2.

Ebert, D. S.; Musgrave, F. K.; Peachey, D.; Perlin, K. and Worley, S. (2003). *A Procedural Approach*. Elsevier BV, xx–xxiii. doi: 10.1016/b978-155860848-1/50029-2.

Efros, A. A. and Freeman, W. T. (2001). "Image Quilting for Texture Synthesis and Transfer". *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*. Association for Computing Machinery (ACM). doi: 10.1145/383259.383296.

Efros, A. A. and Leung, T. K. (1999). "Texture Synthesis by Non-Parametric Sampling". *Proceedings of the International Conference on Computer Vision*. Corfu, Greece, 1033–1038.

Eggers, G. and Renap, K. (2007). *Method and Apparatus for Automatic Support Generation for an Object Made by Means of a Rapid Prototype Production Method*. US Patent 20100228369, Materialize.

Eisenacher, C.; Tappan, C.; Burley, B.; Teece, D. and Shek, A. (2010). "Example-Based Texture Synthesis on Disney's Tangled". *ACM SIGGRAPH 2010 Talks on - SIGGRAPH '10*. Association for Computing Machinery (ACM). doi: 10.1145/1837026.1837068.

Eschenauer, H. A.; Kobelev, V. V. and Schumacher, A. (1994). "Bubble Method for Topology and Shape Optimization of Structures". *Structural Optimization* 8.1, 42–51. doi: 10.1007/bf01742933.

Eschenauer, H. A. and Olhoff, N. (2001). "Topology Optimization of Continuum Structures: A Review". *Applied Mechanics Reviews* 54.4, 331. doi: 10.1115/1.1388075.

Ezair, B.; Massarwi, F. and Elber, G. (2015). "Orientation Analysis of 3D Objects Toward Minimal Support Volume in 3D-Printing". *Computers & Graphics* 51, 117–124. doi: 10.1016/j.cag.2015.05.009.

Fan, C.; Luo, J.; Liu, J. and Xu, Y. (2011). "Half-Plane Voronoi Diagram". *2011 Eighth International Symposium on Voronoi Diagrams in Science and Engineering*. Institute of Electrical & Electronics Engineers (IEEE). doi: 10.1109/isvd.2011.25.

Fortune, S. (1987). "A Sweepline Algorithm for Voronoi Diagrams". *Algorithmica* 2.1-4, 153–174. eprint: http://link.springer.com/content/pdf/10.1007/BF01840357.pdf.

Frank, D. and Fadel, G. (1995). "Expert System-Based Selection of the Preferred Direction of Build for Rapid Prototyping Processes". *Journal of Intelligent Manufacturing* 6.5, 339–345.

Frick, U.; Mele, T. V. and Block, P. (2015). "Decomposing Three-Dimensional Shapes Into Self-Supporting, Discrete-Element Assemblies". *Modelling Behaviour*. Springer Science + Business Media, 187–201. doi: 10.1007/978-3-319-24208-8_16.

Fryazinov, O.; Sanchez, M. and Pasko, A. (2015). "Shape Conforming Volumetric Interpolation With Interior Distances". *Computers & Graphics* 46, 149–155. doi: 10.1016/j.cag.2014.09.028.

Fryazinov, O.; Vilbrandt, T. and Pasko, A. (2013). "Multi-Scale Space-Variant FRep Cellular Structures". *Computer-Aided Design* 45.1, 26–34. doi: 10.1016/j.cad.2011.09.007.

Gagnon, J.; Dagenais, F. and Paquette, E. (2016). "Dynamic Lapped Texture for Fluid Simulations". *The Visual Computer* 32.6-8, 901–909. doi: 10.1007/s00371-016-1262-8.

Gaitanaros, S.; Kyriakides, S. and Kraynik, A. M. (2012). "On the Crushing Response of Random Open-Cell Foams". *Int. J. Solids Struct.* 49.19–20, 2733–2743.

Gal, R.; Wexler, Y.; Ofek, E.; Hoppe, H. and Cohen-Or, D. (2010). "Seamless Montage for Texturing Models". *Computer Graphics Forum* 29.2, 479–486. doi: 10.1111/j.1467-8659.2009.01617.x.

Gallego, G. and Yezzi, A. (2015). "A Compact Formula for the Derivative of a 3-D Rotation in Exponential Coordinates". *Journal of Mathematical Imaging and Vision* 51.3, 378–384. doi: 10.1007/s10851-014-0528-x.

Gao, H.; Zhu, J.; Zhang, W. and Zhou, Y. (2015a). "An Improved Adaptive Constraint Aggregation for Integrated Layout and Topology Optimization". *Computer Methods in Applied Mechanics and Engineering* 289, 387–408. doi: 10.1016/j.cma.2015.02.022.

Gao, W. et al. (2015b). "The Status, Challenges, and Future of Additive Manufacturing in Engineering". *Computer-Aided Design* 69, 65–89. eprint: https://engineering.purdue.edu/ZhangLab/publications/papers/2015-cad-review.pdf.

Garg, A.; Sageman-Furnas, A. O.; Deng, B.; Yue, Y.; Grinspun, E.; Pauly, M. and Wardetzky, M. (2014). "Wire Mesh Design". *ACM Trans. Graph.* 33.4, 66–1. eprint: https://pdfs.semanticscholar.org/20d2/d26b31be7358abee509091e76648aba01767.pdf.

Gaynor, A. T.; Meisel, N. A.; Williams, C. B. and Guest, J. K. (2014). "Topology Optimization for Additive Manufacturing: Considering Maximum Overhang Constraint". *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics (AIAA). doi: 10.2514/6.2014-2036.

Gibson, I.; Rosen, D. and Stucker, B. (2014). *Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*. Springer. eprint: http://www.technology.matthey.com/wp-content/uploads/pdf/173-289-pmr-jul15.pdf#page=23.

Gibson, L. J. and Ashby, M. F. (1997). *Cellular Solids: Structure and Properties*. Cambridge university press.

Glauser, O.; Vartok, B.; Ma, W.; Panozzo, D.; Jacobson, A.; Hilliges, O. and Sorkine-Hornung, O. (2016). "Rig Animation With a Tangible and Modular Input Device". *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16 Adjunct*. Association for Computing Machinery (ACM). doi: 10.1145/2984751.2985696.

Grassia, F. S. (1998). "Practical Parameterization of Rotations Using the Exponential Map". *Journal of Graphics Tools* 3.3, 29–48. doi: 10.1080/10867651.1998.10487493.

Guennebaud, G.; Jacob, B., et al. (2010). *Eigen v3*. http://eigen.tuxfamily.org.

Guessasma, S.; Zhang, W.; Zhu, J.; Belhabib, S. and Nouri, H. (2015). "Challenges of Additive Manufacturing Technologies From an Optimisation Perspective". *International Journal for Simulation and Multidisciplinary Design Optimization* 6, A9. doi: 10.1051/smdo/2016001.

Guest, J. K.; Prévost, J. H. and Belytschko, T. (2004). "Achieving Minimum Length Scale in Topology Optimization Using Nodal Design Variables and Projection Functions". *International Journal for Numerical Methods in Engineering* 61.2, 238–254. doi: 10.1002/nme.1064.

Guest, J. K. (2015). "Optimizing the Layout of Discrete Objects in Structures and Materials: A Projection-Based Topology Optimization Approach". *Computer Methods in Applied Mechanics and Engineering* 283, 330–351. doi: 10.1016/j.cma.2014.09.006.

Guo, X.; Zhang, W. and Zhong, W. (2014a). "Doing Topology Optimization Explicitly and Geometrically—A New Moving Morphable Components Based Framework". *Journal of Applied Mechanics* 81.8, 081009. doi: 10.1115/1.4027609.

Guo, X.; Zhang, W. and Zhong, W. (2014b). "Explicit Feature Control in Structural Topology Optimization via Level Set Method". *Computer Methods in Applied Mechanics and Engineering* 272, 354–378. doi: 10.1016/j.cma.2014.01.010.

Ha, S. and Guest, J. K. (2014). "Optimizing Inclusion Shapes and Patterns in Periodic Materials Using Discrete Object Projection". *Structural and Multidisciplinary Optimization* 50.1, 65–80. doi: 10.1007/s00158-013-1026-2.

Haber, R. B.; Jog, C. S. and Bendsøe, M. P. (1996). "A New Approach to Variable-Topology Shape Design Using a Constraint on Perimeter". *Structural Optimization* 11.1-2, 1–12. doi: 10.1007/bf01279647.

Han, C.; Risser, E.; Ramamoorthi, R. and Grinspun, E. (2008). "Multiscale Texture Synthesis". *ACM SIGGRAPH 2008 papers on - SIGGRAPH '08*. Association for Computing Machinery (ACM). doi: 10.1145/1399504.1360650.

Han, Z.; Liu, Z.; Han, J. and Bu, S. (2014). "3D Shape Creation by Style Transfer". *The Visual Computer* 31.9, 1147–1161. doi: 10.1007/s00371-014-0999-1.

Hao, J.; Fang, L. and Williams, R. E. (2011). "An Efficient Curvature-based Partitioning of Large-scale STL Models". *Rapid Prototyping Journal* 17.2, 116–127. doi: 10.1108/13552541111113862.

Harker, J. (2011). *Crania Anatomica Filigre: Me to You*. https://www.kickstarter.com/projects/joshharker/crania-anatomica-filigre-me-to-you.

Hart, G. W. (2008). "Sculptural Forms From Hyperbolic Tessellations". *2008 IEEE International Conference on Shape Modeling and Applications*. Institute of Electrical & Electronics Engineers (IEEE). doi: 10.1109/smi.2008.4547963.

Hašan, M.; Fuchs, M.; Matusik, W.; Pfister, H. and Rusinkiewicz, S. (2010). "Physical Reproduction of Materials With Specified Subsurface Scattering". *ACM SIGGRAPH 2010 papers on - SIGGRAPH '10*. Association for Computing Machinery (ACM). doi: 10.1145/1833349.1778798.

Haumont, D.; Debeir, O. and Sillion, F. (2003). "Volumetric Cell-And-Portal Generation". *Computer Graphics Forum* 22.3, 303–312. doi: 10.1111/1467-8659.00677.

Hecht, F. (2012). "New Development in FreeFem++". *Journal of Numerical Mathematics* 20.3-4. doi: 10.1515/jnum-2012-0013.

Heide, E. (2011). *Method for Generating and Building Support Structures With Deposition-Based Digital Manufacturing Systems*. Patent. US Patent 20110178621 A1.

Hergel, J. and Lefebvre, S. (2014). "Clean Color: Improving Multi-Filament 3D Prints". *Computer Graphics Forum* 33.2, 469–478. doi: 10.1111/cgf.12318.

Hergel, J. and Lefebvre, S. (2015). "3D Fabrication of 2D Mechanisms". *Computer Graphics Forum* 34.2, 229–238. doi: 10.1111/cgf.12555.

Herholz, P.; Matusik, W. and Alexa, M. (2015). "Approximating Free-Form Geometry With Height Fields for Manufacturing". *Computer Graphics Forum* 34.2, 239–251. doi: 10.1111/cgf.12556.

Hertzmann, A.; Jacobs, C. E.; Oliver, N.; Curless, B. and Salesin, D. H. (2001). "Image Analogies". *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: ACM, 327–340. doi: 10.1145/383259.383295.

Hildebrand, K.; Bickel, B. and Alexa, M. (2012). "Crdbrd: Shape Fabrication by Sliding Planar Slices". *Computer Graphics Forum* 31.2pt3, 583–592. doi: 10.1111/j.1467-8659.2012.03037.x.

Hildebrand, K.; Bickel, B. and Alexa, M. (2013). "Orthogonal Slicing for Additive Manufacturing". *Computers & Graphics* 37.6, 669–675. doi: 10.1016/j.cag.2013.05.011.

Holmberg, E.; Torstenfelt, B. and Klarbring, A. (2013). "Stress Constrained Topology Optimization". *Structural and Multidisciplinary Optimization* 48.1, 33–47. doi: 10.1007/s00158-012-0880-7.

Hornus, S.; Lefebvre, S.; Dumas, J. and Claux, F. (2015). *Tight Printable Enclosures for Additive Manufacturing*. Research Report RR-8712. Inria, 22.

Hornus, S.; Lefebvre, S.; Dumas, J. and Claux, F. (2016). "Tight Printable Enclosures and Support Structures for Additive Manufacturing". *Eurographics Workshop on Graphics for Digital Fabrication*. The Eurographics Association. doi: 10.2312/gdf.20161074.

Hsu, S. and Keyser, J. (2010). "Piles of Objects". *ACM SIGGRAPH Asia 2010 papers on - SIGGRAPH ASIA '10*. Association for Computing Machinery (ACM). doi: 10.1145/1882262.1866181.

Hu, K.; Jin, S. and Wang, C. C. L. (2015). "Support Slimming for Single Material Based Additive Manufacturing". *Computer-Aided Design* 65, 1–10. doi: 10.1016/j.cad.2015.03.001.

Hu, R.; Li, H.; Zhang, H. and Cohen-Or, D. (2014). "Approximate Pyramidal Shape Decomposition". *TOG* 33.6, 1–12. doi: 10.1145/2661229.2661244.

Huang, P.; Wang, C. C. and Chen, Y. (2014a). *Algorithms for Layered Manufacturing in Image Space*. ASME Press.

Huang, Q.; Guibas, L. J. and Mitra, N. J. (2014b). "Near-Regular Structure Discovery Using Linear Programming". *ACM Transactions on Graphics* 33.3, 1–17. doi: 10.1145/2535596.

Huang, X.; Ye, C.; Mo, J. and Liu, H. (2009a). "Slice Data Based Support Generation Algorithm for Fused Deposition Modeling". *Tsinghua Science and Technology* 14.S1, 223–228. doi: 10.1016/s1007-0214(09)70096-3.

Huang, X.; Ye, C.; Wu, S.; Guo, K. and Mo, J. (2009b). "Sloping Wall Structure Support Generation for Fused Deposition Modeling". *The International Journal of Advanced Manufacturing Technology* 42.11-12, 1074–1081. doi: 10.1007/s00170-008-1675-2.

Hull, C. W. (1984). "Apparatus for Production of Three-Dimensional Objects by Stereolithography". US Patent 4,575,330. url: https://www.google.com/patents/us4575330.

Hunter, W. (2009). "Predominantly Solid-Void Three-Dimensional Topology Optimisation Using Open Source Software". Doctoral dissertation. Stellenbosch: University of Stellenbosch. eprint: http://scholar.sun.ac.za/bitstream/handle/10019.1/2648/Hunter,+W.pdf?sequence=1.

Hurtut, T.; Landes, P.
bibinitperiod; Thollot, J.; Gousseau, Y.; Drouillhet, R. and Coeurjolly, J.
bibinitperiod (2009). "Appearance-Guided Synthesis of Element Arrangements by Example". *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering - NPAR '09*. Association for Computing Machinery (ACM). doi: 10.1145/1572614.1572623.

Ijiri, T.; Mêch, R.; Igarashi, T. and Miller, G. (2008). "An Example-Based Procedural System for Element Arrangement". *Computer Graphics Forum* 27.2, 429–436. doi: 10.1111/j.1467-8659.2008.01140.x.

Ion, A.; Frohnhofen, J.; Wall, L.; Kovacs, R.; Alistar, M.; Lindsay, J.; Lopes, P.; Chen, H. and Baudisch, P. (2016). "Metamaterial Mechanisms". *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*. Association for Computing Machinery (ACM). doi: 10.1145/2984511.2984540.

Jagnow, R.; Dorsey, J. and Rushmeier, H. (2004). "Stereological Techniques for Solid Textures". *ACM SIGGRAPH 2004 Papers on - SIGGRAPH '04*. Association for Computing Machinery (ACM). doi: 10.1145/1186562.1015724.

Jakus, A. E. et al. (2016). "Hyperelastic "Bone": A Highly Versatile, Growth Factor-Free, Osteoregenerative, Scalable, and Surgically Friendly Biomaterial". *Science Translational Medicine* 8.358, 358ra127–358ra127. doi: 10.1126/scitranslmed.aaf7704.

Jansen, M.; Lazarov, B. S.; Schevenels, M. and Sigmund, O. (2013). "On the Similarities Between Micro/Nano Lithography and Topology Optimization Projection Methods". *Structural and Multidisciplinary Optimization* 48.4, 717–730. doi: 10.1007/s00158-013-0941-6.

Jin, Y.; He, Y. and Fu, J. (2015). "Support Generation for Additive Manufacturing Based on Sliced Data". *The International Journal of Advanced Manufacturing Technology* 80.9-12, 2041–2052. doi: 10.1007/s00170-015-7190-3.

Jog, C. S. and Haber, R. B. (1996). "Stability of Finite Element Models for Distributed-Parameter Optimization and Topology Design". *Computer Methods in Applied Mechanics and Engineering* 130.3-4, 203–226. doi: 10.1016/0045-7825(95)00928-0.

Johnson, S. G. (2016). *The NLopt Nonlinear-Optimization Package*.

Jones, B.; Thuerey, N.; Shinar, T. and Bargteil, A. W. (2016). "Example-Based Plastic Deformation of Rigid Bodies". *ACM Transactions on Graphics* 35.4, 1–11. doi: 10.1145/2897824.2925979.

Jones, B.; Ward, S.; Jallepalli, A.; Perenia, J. and Bargteil, A. W. (2014). "Deformation Embedding for Point-Based Elastoplastic Simulation". *ACM Transactions on Graphics* 33.2, 1–9. doi: 10.1145/2560795.

Kalojanov, J.; Wand, M. and Slusallek, P. (2016). "Building Construction Sets by Tiling Grammar Simplification". *Computer Graphics Forum* 35.2, 13–25. doi: 10.1111/cgf.12807.

Kang, Z.; Wang, Y. and Wang, Y. (2016). "Structural Topology Optimization With Minimum Distance Control of Multiphase Embedded Components by Level Set Method". *Computer Methods in Applied Mechanics and Engineering* 306, 299–318. doi: 10.1016/j.cma.2016.04.001.

Kang, Z. and Wang, Y. (2013). "Integrated Topology Optimization With Embedded Movable Holes Based on Combined Description by Material Density and Level Sets". *Computer Methods in Applied Mechanics and Engineering* 255, 1–13. doi: 10.1016/j.cma.2012.11.006.

Kaspar, A.; Neubert, B.; Lischinski, D.; Pauly, M. and Kopf, J. (2015). "Self Tuning Texture Optimization". *Computer Graphics Forum* 34.2, 349–359. doi: 10.1111/cgf.12565.

Kaufman, D. M.; Sueda, S.; James, D. L. and Pai, D. K. (2008). "Staggered Projections for Frictional Contact in Multibody Systems". *ACM Transactions on Graphics* 27.5, 1. doi: 10.1145/1409060.1409117.

Kim, T. (2015). "Quaternion Julia Set Shape Optimization". *Computer Graphics Forum* 34.5, 167–176. doi: 10.1111/cgf.12705.

Kodama, H. (1981). "Automatic Method for Fabricating a Three-Dimensional Plastic Model With Photo-Hardening Polymer". *Review of Scientific Instruments* 52.11, 1770. doi: 10.1063/1.1136492.

Kopf, J.; Fu, C.; Cohen-Or, D.; Deussen, O.; Lischinski, D. and Wong, T. (2007). "Solid Texture Synthesis From 2D Exemplars". *ACM Transactions on Graphics* 26.3, 2. doi: 10.1145/1276377.1276380.

Kosaka, I. and Swan, C. C. (1999). "A Symmetry Reduction Method for Continuum Structural Topology Optimization". *Computers & Structures* 70.1, 47–61. doi: 10.1016/s0045-7949(98)00158-8.

Kou, X. Y. and Tan, S. T. (2012). "Microstructural Modelling of Functionally Graded Materials Using Stochastic Voronoi Diagram and B-Spline Representations". *International Journal of Computer Integrated Manufacturing* 25.2, 177–188. doi: 10.1080/0951192x.2011.627948.

Koyama, Y.; Takayama, K.; Umetani, N. and Igarashi, T. (2012). "Real-Time Example-Based Elastic Deformation". *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*. Eurographics Association, 19–24. eprint: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.295.1214\&rep=rep1\&type=pdf.

Kritchman, E.; Gothait, H. and Miller, G. (2008). *System and Method for Printing and Supporting Three Dimensional Objects*. Patent. US Patent 7364686.

Kulkarni, P.; Marsan, A. and Dutta, D. (2000). "A Review of Process Planning Techniques in Layered Manufacturing". *Rapid Prototyping Journal* 6.1, 18–35. doi: 10.1108/13552540010309859.

Kwatra, V.; Essa, I.; Bobick, A. and Kwatra, N. (2005). "Texture Optimization for Example-Based Synthesis". *ACM Transactions on Graphics* 24.3, 795. doi: 10.1145/1073204.1073263.

Kwatra, V.; Schödl, A.; Essa, I.; Turk, G. and Bobick, A. (2003). "Graphcut Textures: Image and Video Synthesis Using Graph Cuts". *ACM SIGGRAPH 2003 Papers on - SIGGRAPH '03*. Association for Computing Machinery (ACM). doi: 10.1145/1201775.882264.

Lagae, A.; Dumont, O. and Dutre, P. (2005). "Geometry Synthesis by Example". *International Conference on Shape Modeling and Applications 2005 (SMI'05)*. IEEE, 174–183. eprint: http://www.academia.edu/download/33937747/LDD05GSE.pdf.

Lagae, A.; Lefebvre, S.; Cook, R.; DeRose, T.; Drettakis, G.; Ebert, D. S.; Lewis, J. P.; Perlin, K. and Zwicker, M. (2010). "A Survey of Procedural Noise Functions". *Computer Graphics Forum* 29.8.

Lagae, A.; Lefebvre, S.; Drettakis, G. and Dutré, P. (2009). "Procedural Noise Using Sparse Gabor Convolution". *ACM Trans. Graph.* 28.3, 54:1–54:10.

Lan, Y.; Dong, Y.; Pellacini, F. and Tong, X. (2013). "Bi-Scale Appearance Fabrication". *ACM Transactions on Graphics* 32.4, 1. doi: 10.1145/2461912.2461989.

Landes, P.; Galerne, B. and Hurtut, T. (2013). "A Shape-Aware Model for Discrete Texture Synthesis". *Computer Graphics Forum* 32.4, 67–76. doi: 10.1111/cgf.12152.

Landes, P. and Soler, C. (2009). "Content-Aware Texture Synthesis". Doctoral dissertation. INRIA. eprint: https://hal.archives-ouvertes.fr/docs/00/39/42/62/PDF/RR-6959.pdf.

Langelaar, M. (2016a). "An Additive Manufacturing Filter for Topology Optimization of Print-Ready Designs". *Structural and Multidisciplinary Optimization*. doi: 10.1007/s00158-016-1522-2.

Langelaar, M. (2016b). "Topology Optimization of 3D Self-Supporting Structures for Additive Manufacturing". *Additive Manufacturing*. doi: 10.1016/j.addma.2016.06.010.

Langlois, T.; Shamir, A.; Dror, D.; Matusik, W. and Levin, D. I. W. (2016). "Stochastic Structural Analysis for Context-Aware Design and Fabrication". *ACM Transactions on Graphics* 35.6, 1–13. doi: 10.1145/2980179.2982436.

Larsen, U. D.; Sigmund, O. and Bouwstra, S. (1997). "Design and Fabrication of Compliant Micromechanisms and Structures With Negative Poisson's Ratio". *Microelectromechanical Systems, Journal of* 6.2, 99–106. doi: 10.1109/memsys.1996.494009.

Lazarov, B. S. and Sigmund, O. (2011). "Filters in Topology Optimization Based on Helmholtz-Type Differential Equations". *International Journal for Numerical Methods in Engineering* 86.6, 765–781. doi: 10.1002/nme.3072.

Lazarov, B. S. (2014). "Topology Optimization Using Multiscale Finite Element Method for High-Contrast Media". *Large-Scale Scientific Computing*. Springer Science + Business Media, 339–346. doi: 10.1007/978-3-662-43880-0_38.

Lazarov, B. S.; Schevenels, M. and Sigmund, O. (2012a). "Topology Optimization Considering Material and Geometric Uncertainties Using Stochastic Collocation Methods". *Structural and Multidisciplinary Optimization* 46.4, 597–612. doi: 10.1007/s00158-012-0791-7.

Lazarov, B. S.; Schevenels, M. and Sigmund, O. (2012b). "Topology Optimization With Geometric Uncertainties by Perturbation Techniques". *International Journal for Numerical Methods in Engineering* 90.11, 1321–1336. doi: 10.1002/nme.3361.

Lazarov, B. S.; Wang, F. and Sigmund, O. (2016). "Length Scale and Manufacturability in Density-Based Topology Optimization". *Archive of Applied Mechanics* 86.1-2, 189–218. doi: 10.1007/s00419-015-1106-4.

Leary, M.; Merli, L.; Torti, F.; Mazur, M. and Brandt, M. (2014). "Optimal Topology for Additive Manufacture: A Method for Enabling Additive Manufacture of Support-Free Optimal Structures". *Materials & Design* 63, 678–690. doi: 10.1016/j.matdes.2014.06.015.

Lee, E.; James, K. A. and Martins, J. R. R. A. (2012a). "Stress-Constrained Topology Optimization With Design-Dependent Loading". *Structural and Multidisciplinary Optimization* 46.5, 647–661. doi: 10.1007/s00158-012-0780-x.

Lee, J. and Lee, K. (2016). "Block-Based Inner Support Structure Generation Algorithm for 3D Printing Using Fused Deposition Modeling". *The International Journal of Advanced Manufacturing Technology*. doi: 10.1007/s00170-016-9239-3.

Lee, S.; Park, T.; Kim, J. and Kim, C. (2012b). "Adaptive Synthesis of Distance Fields". *IEEE Transactions on Visualization and Computer Graphics* 18.7, 1135–1145. doi: 10.1109/tvcg.2011.134.

Lefebvre, S. (2013). "IceSL: A GPU Accelerated CSG Modeler and Slicer". *AEFA'13, 18th European Forum on Additive Manufacturing*. eprint: http://webloria.loria.fr/~slefebvr/icesl/icesl-whitepaper.pdf.

Lefebvre, S. (2014). "Runtime By-Example Texture Synthesis". Accreditation to supervise research. Université de Lorraine (Nancy). URL: https://hal.inria.fr/tel-01388378.

Lefebvre, S. (2015). *3D Infilling: Faster, Stronger, Simpler*. URL: http://sylefeb.blogspot.fr/2015/07/3dprint-3d-infilling-faster-stronger.html.

Lefebvre, S. and Hoppe, H. (2005). "Parallel Controllable Texture Synthesis". *ACM SIGGRAPH 2005 Papers on - SIGGRAPH '05*. Association for Computing Machinery (ACM). DOI: 10.1145/1186822.1073261.

Lefebvre, S. and Hoppe, H. (2006). "Appearance-Space Texture Synthesis". *ACM SIGGRAPH 2006 Papers on - SIGGRAPH '06*. Association for Computing Machinery (ACM). DOI: 10.1145/1179352.1141921.

Lefebvre, S.; Hornus, S. and Lasram, A. (2010). "By-Example Synthesis of Architectural Textures". *ACM Transactions on Graphics* 29.4, 1. DOI: 10.1145/1778765.1778821.

Lefebvre, S. and Neyret, F. (2003). "Pattern Based Procedural Textures". *Proceedings of the 2003 symposium on Interactive 3D graphics - SI3D '03*. Association for Computing Machinery (ACM). DOI: 10.1145/641480.641518.

Lempitsky, V. and Ivanov, D. (2007). "Seamless Mosaicing of Image-Based Texture Maps". *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 1–6. DOI: 10.1109/CVPR.2007.383078.

Li, D.; Dai, N.; Jiang, X. and Chen, X. (2015). "Interior Structural Optimization Based on the Density-Variable Shape Modeling of 3D Printed Objects". *The International Journal of Advanced Manufacturing Technology* 83.9-12, 1627–1635. DOI: 10.1007/s00170-015-7704-z.

Li, H.; Zhang, H.; Wang, Y.; Cao, J.; Shamir, A. and Cohen-Or, D. (2013). "Curve Style Analysis in a Set of Shapes". *Computer Graphics Forum* 32.6, 77–88.

Li, Q.; Chen, W.; Liu, S. and Tong, L. (2016). "Structural Topology Optimization Considering Connectivity Constraint". *Structural and Multidisciplinary Optimization*. DOI: 10.1007/s00158-016-1459-5.

Li, S.; Huang, J.; Goes, F. de; Jin, X.; Bao, H. and Desbrun, M. (2014). "Space-Time Editing of Elastic Motion Through Material Optimization and Reduction". *ACM Transactions on Graphics* 33.4, 1–10. DOI: 10.1145/2601097.2601217.

Li, Y.; Bao, F.; Zhang, E.; Kobayashi, Y. and Wonka, P. (2011). "Geometry Synthesis on Surfaces Using Field-Guided Shape Grammars". *IEEE Transactions on Visualization and Computer Graphics* 17.2, 231–243. DOI: 10.1109/tvcg.2010.36.

Lindenmayer, A. (1968a). "Mathematical Models for Cellular Interactions in Development I. Filaments With One-Sided Inputs". *Journal of Theoretical Biology* 18.3, 280–299. DOI: 10.1016/0022-5193(68)90079-9.

Lindenmayer, A. (1968b). "Mathematical Models for Cellular Interactions in Development II. Simple and Branching Filaments With Two-Sided Inputs". *Journal of Theoretical Biology* 18.3, 300–315. DOI: 10.1016/0022-5193(68)90080-5.

Liu, H.; Vimont, U.; Wand, M.; Cani, M.; Hahmann, S.; Rohmer, D. and Mitra, N. J. (2015a). "Replaceable Substructures for Efficient Part-Based Modeling". *Computer Graphics Forum* 34.2, 503–513. DOI: 10.1111/cgf.12579.

Liu, J. and Ma, Y.
bibinitperiod (2015). "3D Level-Set Topology Optimization: A Machining Feature-Based Approach". *Structural and Multidisciplinary Optimization* 52.3, 563–582. DOI: 10.1007/s00158-015-1263-7.

Liu, J. and Ma, Y. (2016). "A Survey of Manufacturing Oriented Topology Optimization Methods". *Advances in Engineering Software* 100, 161–175. DOI: 10.1016/j.advengsoft.2016.07.017.

Liu, J.; Yu, H. and Ma, Y. (2016). "Minimum Void Length Scale Control in Level Set Topology Optimization Subject to Machining Radii". *Computer-Aided Design*. DOI: 10.1016/j.cad.2016.09.007.

Liu, K. and Tovar, A. (2014). "An Efficient 3D Topology Optimization Code Written in Matlab". *Structural and Multidisciplinary Optimization* 50.6, 1175–1196. DOI: 10.1007/s00158-014-1107-x.

Liu, L.; Chambers, E. W.; Letscher, D. and Ju, T. (2010). "A Simple and Robust Thinning Algorithm on Cell Complexes". *Computer Graphics Forum* 29.7, 2253–2260. DOI: 10.1111/j.1467-8659.2010.01814.x.

Liu, L.; Shamir, A.; Wang, C. and Whitening, E. (2014a). "3D Printing Oriented Design: Geometry and Optimization". *SIGGRAPH Asia 2014 Courses on - SA '14*. Association for Computing Machinery (ACM). DOI: 10.1145/2659467.2675050.

Liu, S. and Wang, C. C. L. (2011). "Fast Intersection-Free Offset Surface Generation From Freeform Models With Triangular Meshes". *IEEE Transactions on Automation Science and Engineering* 8.2, 347–360. DOI: 10.1109/tase.2010.2066563.

Liu, S.; Li, Q.; Chen, W.; Tong, L. and Cheng, G. (2015b). "An Identification Method for Enclosed Voids Restriction in Manufacturability Design for Additive Manufacturing Structures". *Frontiers of Mechanical Engineering* 10.2, 126–137. DOI: 10.1007/s11465-015-0340-3.

Liu, T.; Wang, S.; Li, B. and Gao, L. (2014b). "A Level-Set-Based Topology and Shape Optimization Method for Continuum Structure Under Geometric Constraints". *Structural and Multidisciplinary Optimization* 50.2, 253–273. DOI: 10.1007/s00158-014-1045-7.

Liu, X. and Shapiro, V. (2015). "Random Heterogeneous Materials via Texture Synthesis". *Computational Materials Science* 99, 177–189. DOI: 10.1016/j.commatsci.2014.12.017.

Liu, X. and Shapiro, V. (2016). "Homogenization of Material Properties in Additively Manufactured Structures". *Computer-Aided Design* 78, 71–82. DOI: 10.1016/j.cad.2016.05.017.

Liu, Y.; Wang, W.; Lévy, B.; Sun, F.; Yan, D.; Lu, L. and Yang, C. (2009). "On Centroidal Voronoi Tessellation—energy Smoothness and Fast Computation". *ACM Transactions on Graphics* 28.4, 1–17. DOI: 10.1145/1559755.1559758.

Liu, Z.; Korvink, J. and Huang, R. (2005). "Structure Topology Optimization: Fully Coupled Level Set Method via FEMLAB". *Structural and Multidisciplinary Optimization* 29.6, 407–417. DOI: 10.1007/s00158-004-0503-z.

Lu, L.; Lévy, B. and Wang, W. (2012). "Centroidal Voronoi Tessellation of Line Segments and Graphs". *Computer Graphics Forum* 31.2pt4, 775–784. DOI: 10.1111/j.1467-8659.2012.03058.x.

Lu, L. et al. (2014). "Build-To-Last: Strength to Weight 3D Printed Objects". *ACM Transactions on Graphics* 33.4, 1–10. DOI: 10.1145/2601097.2601168.

Luo, L.; Baran, I.; Rusinkiewicz, S. and Matusik, W. (2012). "Chopper: Partitioning Models Into 3D-Printable Parts". *ACM Transactions on Graphics* 31.6, 1. DOI: 10.1145/2366145.2366148.

Luo, S.; Yue, Y.; Huang, C.; Chung, Y.; Imai, S.; Nishita, T. and Chen, B. (2015). "Legolization: Optimizing LEGO Designs". *ACM Transactions on Graphics* 34.6, 1–12. DOI: 10.1145/2816795.2818091.

Luxner, M. H.; Stampfl, J. and Pettermann, H. E. (2007). "Numerical Simulations of 3D Open Cell Structures – Influence of Structural Irregularities on Elasto-Plasticity and Deformation Localization". *International Journal of Solids and Structures* 44.9, 2990–3003. DOI: 10.1016/j.ijsolstr.2006.08.039.

Ma, C.; Huang, H.; Sheffer, A.; Kalogerakis, E. and Wang, R. (2014). "Analogy-Driven 3D Style Transfer". *Computer Graphics Forum* 33.2, 175–184. DOI: 10.1111/cgf.12307.

Ma, C.; Wei, L.; Lefebvre, S. and Tong, X. (2013). "Dynamic Element Textures". *ACM Transactions on Graphics* 32.4, 1. DOI: 10.1145/2461912.2461921.

Ma, C.; Wei, L. and Tong, X. (2011). "Discrete Element Textures". *ACM SIGGRAPH 2011 papers on - SIGGRAPH '11*. Association for Computing Machinery (ACM). DOI: 10.1145/1964921.1964957.

Majhi, J.; Janardan, R.; Smid, M. and Gupta, P. (1999). "On Some Geometric Optimization Problems in Layered Manufacturing". *Computational Geometry* 12.34, 219–239.

Malzbender, T.; Samadani, R.; Scher, S.; Crume, A.; Dunn, D. and Davis, J. (2012). "Printing Reflectance Functions". *ACM Transactions on Graphics* 31.3, 1–11. DOI: 10.1145/2167076.2167078.

MarietHoz, G. and Lefebvre, S. (2014). "Bridges Between Multiple-Point Geostatistics and Texture Synthesis: Review and Guidelines for Future Research". *Computers & Geosciences* 66, 66–80. DOI: 10.1016/j.cageo.2014.01.001.

Martin, S.; Thomaszewski, B.; Grinspun, E. and Gross, M. (2011). "Example-Based Elastic Materials". *ACM SIGGRAPH 2011 papers on - SIGGRAPH '11*. Association for Computing Machinery (ACM). DOI: 10.1145/1964921.1964967.

Martin, T.; Umetani, N. and Bickel, B. (2015). "OmniAD: Data-Driven Omni-Directional Aerodynamics". *ACM Transactions on Graphics* 34.4, 113:1–113:12. DOI: 10.1145/2766919.

Martínez, J.; Dumas, J. and Lefebvre, S. (2016). "Procedural Voronoi Foams for Additive Manufacturing". *ACM Trans. Graph.* 35.4. DOI: 10.1145/2897824.2925922.

Martínez, J.; Dumas, J.; Lefebvre, S. and Wei, L. (2015a). "Structure and Appearance Optimization for Controllable Shape Design". *ACM Trans. Graph.* 34.6, 229:1–229:11. DOI: 10.1145/2816795.2818101.

Martínez, J.; Hornus, S.; Claux, F. and Lefebvre, S. (2015b). "Chained Segment Offsetting for Ray-Based Solid Representations". *Computers & Graphics* 46, 36–47. DOI: 10.1016/j.cag.2014.09.017.

Matusik, W.; Ajdin, B.; Gu, J.; Lawrence, J.; Lensch, H. P. A.; Pellacini, F. and Rusinkiewicz, S. (2009). "Printing Spatially-Varying Reflectance". *ACM SIGGRAPH Asia 2009 papers on - SIGGRAPH Asia '09*. Association for Computing Machinery (ACM). doi: 10.1145/1661412.1618474.

Maule, M.; Comba, J. L.; Torchelsen, R. P. and Bastos, R. (2011). "A Survey of Raster-Based Transparency Techniques". *Computers & Graphics* 35.6, 1023–1034. doi: 10.1016/j.cag.2011.07.006.

McCrae, J.; Singh, K. and Mitra, N. J. (2011). "Slices: A Shape-Proxy Based on Planar Sections". *Proceedings of the 2011 SIGGRAPH Asia Conference on - SA '11*. Association for Computing Machinery (ACM). doi: 10.1145/2024156.2024202.

McCrae, J.; Umetani, N. and Singh, K. (2014). "FlatFitFab: Interactive Modeling With Planar Sections". *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*. Association for Computing Machinery (ACM). doi: 10.1145/2642918.2647388.

Medeiros e Sá, A.; Mello, V. M.; Echavarria, K. R. and Covill, D. (2015). "Adaptive Voids". *The Visual Computer* 31.6-8, 799–808.

Megaro, V.; Thomaszewski, B.; Gauge, D.; Grinspun, E.; Coros, S. and Gross, M. (2014). "Chacra: An Interactive Design System for Rapid Character Crafting". *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 123–130. eprint: https://graphics.ethz.ch/Downloads/Publications/Papers/2014/Meg14a/Meg14a.pdf.

Mei, Y.; Wang, X. and Cheng, G. (2008). "A Feature-Based Topological Optimization for Structure Design". *Advances in Engineering Software* 39.2, 71–87. doi: 10.1016/j.advengsoft.2007.01.023.

Merrell, P. (2007). "Example-Based Model Synthesis". *Proceedings of the 2007 symposium on Interactive 3D graphics and games - I3D '07*. Association for Computing Machinery (ACM). doi: 10.1145/1230100.1230119.

Merrell, P. and Manocha, D. (2008). "Continuous Model Synthesis". *ACM SIGGRAPH Asia 2008 papers on - SIGGRAPH Asia '08*. Association for Computing Machinery (ACM). doi: 10.1145/1457515.1409111.

Merrell, P. and Manocha, D. (2011). "Model Synthesis: A General Procedural Modeling Algorithm". *IEEE Transactions on Visualization and Computer Graphics* 17.6, 715–728. doi: 10.1109/tvcg.2010.112.

Michailidis, G. (2014). "Manufacturing Constraints and Multi-Phase Shape and Topology Optimization via a Level-Set Method". Doctoral dissertation. Ecole Polytechnique X. eprint: https://pastel.archives-ouvertes.fr/pastel-00937306/file/thesis.pdf.

Miguel, E.; Lepoutre, M. and Bickel, B. (2016). "Computational Design of Stable Planar-Rod Structures". *ACM Transactions on Graphics* 35.4, 1–11. doi: 10.1145/2897824.2925978.

Milliez, A.; Wand, M.; Cani, M. and Seidel, H. (2013). "Mutable Elastic Models for Sculpting Structured Shapes". *Computer Graphics Forum* 32.2pt1, 21–30. doi: 10.1111/cgf.12022.

Morgan, H. D.; Cherry, J. A.; Jonnalagadda, S.; Ewing, D. and Sienz, J. (2016). "Part Orientation Optimisation for the Additive Layer Manufacture of Metal

Components". *The International Journal of Advanced Manufacturing Technology*. DOI: 10.1007/s00170-015-8151-6.

Mueller, S.; Im, S.; Gurevich, S.; Teibrich, A.; Pfisterer, L.; Guimbretière, F. and Baudisch, P. (2014a). "WirePrint: 3D Printed Previews for Fast Prototyping". *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*. Association for Computing Machinery (ACM). DOI: 10.1145/2642918.2647359.

Mueller, S.; Kruck, B. and Baudisch, P. (2013). "LaserOrigami: Laser-Cutting 3D Objects". *CHI '13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13*. Association for Computing Machinery (ACM). DOI: 10.1145/2468356.2479544.

Mueller, S.; Mohr, T.; Guenther, K.; Frohnhofen, J. and Baudisch, P. (2014b). "faBrickation: Fast 3D Printing of Functional Objects by Integrating Construction Kit Building Blocks". *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. Association for Computing Machinery (ACM). DOI: 10.1145/2556288.2557005.

Musialski, P.; Auzinger, T.; Birsak, M.; Wimmer, M. and Kobbelt, L. (2015). "Reduced-Order Shape Optimization Using Offset Surfaces". *ACM Transactions on Graphics* 34.4, 102:1–102:9. DOI: 10.1145/2766955.

Musialski, P.; Hafner, C.; Rist, F.; Birsak, M.; Wimmer, M. and Kobbelt, L. (2016). "Non-Linear Shape Optimization Using Local Subspace Projections". *ACM Transactions on Graphics* 35.4, 1–13. DOI: 10.1145/2897824.2925886.

Nehab, D. and Hoppe, H. (2008). "Random-Access Rendering of General Vector Graphics". *ACM Transactions on Graphics* 27.5, 1. DOI: 10.1145/1409060.1409088.

Nobel-Jørgensen, M.; Aage, N.; Christiansen, A. N.; Igarashi, T.; Bærentzen, J. A. and Sigmund, O. (2015). "3D Interactive Topology Optimization on Hand-Held Devices". *Structural and Multidisciplinary Optimization* 51.6, 1385–1391. DOI: 10.1007/s00158-014-1214-8.

Nocedal, J. and Wright, S. (2006). *Numerical Optimization*. Springer Science & Business Media. eprint: http://thuvien.due.udn.vn:8080/dspace/bitstream/TVDHKT/18432/1/72.pdf.

Norato, J. A.; Bell, B. and Tortorelli, D. (2015). "A Geometry Projection Method for Continuum-Based Topology Optimization With Discrete Elements". *Computer Methods in Applied Mechanics and Engineering* 293, 306–327. DOI: 10.1016/j.cma.2015.05.005.

Oropallo, W. and Piegl, L. A. (2015). "Ten Challenges in 3D Printing". *Engineering with Computers* 32.1, 135–148. DOI: 10.1007/s00366-015-0407-0.

Osher, S. J. and Santosa, F. (2001). "Level Set Methods for Optimization Problems Involving Geometry and Constraints". *Journal of Computational Physics* 171.1, 272–288. DOI: 10.1006/jcph.2001.6789.

Osher, S. and Sethian, J. A. (1988). "Fronts Propagating With Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations". *Journal of Computational Physics* 79.1, 12–49. DOI: 10.1016/0021-9991(88)90002-2.

Overvelde, J. T. (2012). "The Moving Node Approach in Topology Optimization". Doctoral dissertation. TU Delft, Delft University of Technology. eprint: http:

//repository.tudelft.nl/assets/uuid:86c056d8-f368-4239-893f-07ca3a22e112/EM_2012_006_-_Overvelde_-_MSc_-_Report.pdf.

Öztireli, A. C. and Gross, M. (2012). "Analysis and Synthesis of Point Distributions Based on Pair Correlation". *ACM Transactions on Graphics* 31.6, 1. DOI: 10.1145/2366145.2366189.

Pagés, R.; Berjón, D.; Morán, F. and García, N. (2014). "Seamless, Static Multi-Texturing of 3D Meshes". *Computer Graphics Forum* 34.1, 228–238. DOI: 10.1111/cgf.12508.

Pandey, P. M.; Reddy, N. V. and Dhande, S. G. (2003). "Slicing Procedures in Layered Manufacturing: A Review". *Rapid Prototyping Journal* 9.5, 274–288. DOI: 10.1108/13552540310502185.

Panetta, J.; Zhou, Q.; Malomo, L.; Pietroni, N.; Cignoni, P. and Zorin, D. (2015). "Elastic Textures for Additive Fabrication". *ACM Transactions on Graphics* 34.4, 135:1–135:12. DOI: 10.1145/2766937.

Panozzo, D.; Block, P. and Sorkine-Hornung, O. (2013). "Designing Unreinforced Masonry Models". *ACM Transactions on Graphics* 32.4, 1. DOI: 10.1145/2461912.2461958.

Panozzo, D.; Diamanti, O.; Paris, S.; Tarini, M.; Sorkine, E. and Sorkine-Hornung, O. (2015). "Texture Mapping Real-World Objects With Hydrographics". *Computer Graphics Forum* 34.5, 65–75. DOI: 10.1111/cgf.12697.

Papas, M.; Regg, C.; Jarosz, W.; Bickel, B.; Jackson, P.; Matusik, W.; Marschner, S. and Gross, M. (2013). "Fabricating Translucent Materials Using Continuous Pigment Mixtures". *ACM Transactions on Graphics* 32.4, 1. DOI: 10.1145/2461912.2461974.

Paris, J.; Muinos, I.; Navarrina, F.; Colominas, I. and Casteleiro, M. (2005). "A Minimum Weight FEM Formulation for Structural Topological Optimization With Local Stress Constraints". *VI World Congress on Structural and Multidisciplinary Optimization WCSMO6, Rio de Janeiro, Brasil*. Citeseer. eprint: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.2567\&rep=rep1\&type=pdf.

Park, S.; Crawford, R. H. and Beaman, J. J. (2000). "Functionally Gradient Material Representation by Volumetric Multi-Texturing for Solid Freeform Fabrication". *11th Annual Solid Freeform Fabrication Symposium*. eprint: http://sffsymposium.engr.utexas.edu/Manuscripts/2000/2000-43-Park.pdf.

Pasko, A.; Fryazinov, O.; Vilbrandt, T.; Fayolle, P. and Adzhiev, V. (2011). "Procedural Function-Based Modelling of Volumetric Microstructures". *Graphical Models* 73.5, 165–181. DOI: 10.1016/j.gmod.2011.03.001.

Passos, V. A. d.; Walter, M. and Sousa, M. C. (2010). "Sample-Based Synthesis of Illustrative Patterns". *2010 18th Pacific Conference on Computer Graphics and Applications*. Institute of Electrical & Electronics Engineers (IEEE). DOI: 10.1109/pacificgraphics.2010.22.

Peachey, D. R. (1985). "Solid Texturing of Complex Surfaces". *Proceedings of the 12th annual conference on Computer graphics and interactive techniques - SIGGRAPH '85*. Association for Computing Machinery (ACM). DOI: 10.1145/325334.325246.

Pedersen, N. L. (2000). "Maximization of Eigenvalues Using Topology Optimization". *Structural and Multidisciplinary Optimization* 20.1, 2–11. doi: 10.1007/s001580050130.

Pérez, J.; Thomaszewski, B.; Coros, S.; Bickel, B.; Canabal, J. A.; Sumner, R. and Otaduy, M. A. (2015). "Design and Fabrication of Flexible Rod Meshes". *ACM Transactions on Graphics* 34.4, 138:1–138:12. doi: 10.1145/2766998.

Perlin, K. and Hoffert, E. M. (1989). "Hypertexture". *ACM SIGGRAPH Computer Graphics* 23.3, 253–262. doi: 10.1145/74334.74359.

Perlin, K. (1985). "An Image Synthesizer". *ACM SIGGRAPH Computer Graphics* 19.3, 287–296. doi: 10.1145/325165.325247.

Pietroni, N.; Cignoni, P.; Otaduy, M. and Scopigno, R. (2010). "Solid-Texture Synthesis: A Survey". *IEEE Computer Graphics and Applications* 30.4, 74–89. doi: 10.1109/mcg.2009.153.

Pietroni, N.; Tonelli, D.; Puppo, E.; Froli, M.; Scopigno, R. and Cignoni, P. (2015). "Statics Aware Grid Shells". *Computer Graphics Forum* 34.2, 627–641. doi: 10.1111/cgf.12590.

Pintus, R.; Gobbetti, E.; Cignoni, P. and Scopigno, R. (2010). "Shape Enhancement for Rapid Prototyping". *The Visual Computer* 26.6-8, 831–840. doi: 10.1007/s00371-010-0488-0.

Praun, E.; Finkelstein, A. and Hoppe, H. (2000). "Lapped Textures". *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*. Association for Computing Machinery (ACM). doi: 10.1145/344779.344987.

Prévost, R.; Whiting, E.; Lefebvre, S. and Sorkine-Hornung, O. (2013). "Make It Stand: Balancing Shapes for 3D Fabrication". *ACM Transactions on Graphics* 32.4, 1. doi: 10.1145/2461912.2461957.

Prusinkiewicz, P.; Lindenmayer, A. and Hanan, J. (1988). "Development Models of Herbaceous Plants for Computer Imagery Purposes". *ACM SIGGRAPH Computer Graphics* 22.4, 141–150. doi: 10.1145/378456.378503.

Qin, X. and Yang, Y. (2007). "Aura 3D Textures". *IEEE Transactions on Visualization and Computer Graphics* 13.2, 379–389. doi: 10.1109/tvcg.2007.31.

Radman, A.; Huang, X. and Xie, Y. M. (2013). "Topology Optimization of Functionally Graded Cellular Materials". *Journal of Materials Science* 48.4, 1503–1510. doi: 10.1007/s10853-012-6905-1.

Ramanarayanan, G. and Bala, K. (2007). "Constrained Texture Synthesis via Energy Minimization". *IEEE Transactions on Visualization and Computer Graphics* 13.1, 167–178. doi: 10.1109/tvcg.2007.4.

Ray, N.; Neiger, T.; Lévy, B. and Cavin, X. (2005). *Vector Texture Maps on the GPU*. Tech. rep. INRIA - ALICE. eprint: http://alice.loria.fr/publications/papers/2005/VTM/vtm.pdf.

Reiner, T.; Carr, N.; Měch, R.; Šťava, O.; Dachsbacher, C. and Miller, G. (2014). "Dual-Color Mixing for Fused Deposition Modeling Printers". *Computer Graphics Forum* 33.2, 479–486. doi: 10.1111/cgf.12319.

Roberts, A. P. and Garboczi, E. J. (2002). "Elastic Properties of Model Random Three-Dimensional Open-Cell Solids". *Journal of the Mechanics and Physics of Solids* 50.1, 33–55. doi: 10.1016/s0022-5096(01)00056-4.

Rohmer, D.; Hahmann, S. and Cani, M. (2015). "Real-Time Continuous Self-Replicating Details for Shape Deformation". *Computers & Graphics* 51, 67–73. doi: 10.1016/j.cag.2015.05.011.

Rojas-Labanda, S. and Stolpe, M. (2015a). "Automatic Penalty Continuation in Structural Topology Optimization". *Structural and Multidisciplinary Optimization* 52.6, 1205–1221. doi: 10.1007/s00158-015-1277-1.

Rojas-Labanda, S. and Stolpe, M. (2015b). "Benchmarking Optimization Solvers for Structural Topology Optimization". *Structural and Multidisciplinary Optimization* 52.3, 527–547. doi: 10.1007/s00158-015-1250-z.

Rosenkrantz, J. and Louis-Rosenberg, J. (2007). *Nervous System Inc.* http://n-e-r-v-o-u-s.com/about_us.php.

Roveri, R.; Öztireli, A. C.; Martin, S.; Solenthaler, B. and Gross, M. (2015). "Example Based Repetitive Structure Synthesis". *Computer Graphics Forum* 34.5, 39–52. doi: 10.1111/cgf.12695.

Rozvany, G. I. N. (2001). "Aims, Scope, Methods, History and Unified Terminology of Computer-Aided Topology Optimization in Structural Mechanics". *Structural and Multidisciplinary Optimization* 21.2, 90–108. doi: 10.1007/s001580050174.

Rozvany, G. I. N. (2009). "A Critical Review of Established Methods of Structural Topology Optimization". *Structural and Multidisciplinary Optimization* 37.3, 217–237. doi: 10.1007/s00158-007-0217-0.

Rozvany, G. I. N.; Zhou, M. and Birker, T. (1992). "Generalized Shape Optimization Without Homogenization". *Structural Optimization* 4.3-4, 250–252. doi: 10.1007/bf01742754.

Rvachev, V. L. (1982). "Theory of R-Functions and Some Applications". *Naukova Dumka* 552. (In Russian).

Sakurai, K. and Miyata, K. (2014). "Modelling of Non-Periodic Aggregates Having a Pile Structure". *Computer Graphics Forum* 33.1, 190–198. doi: 10.1111/cgf.12266.

Sanchez-Palencia, E. (1980). "Homogenization in Elasticity and Electromagnetism". English. *Non-Homogeneous Media and Vibration Theory*. Vol. 127. Lecture Notes in Physics. Springer Berlin Heidelberg, 84–128. doi: 10.1007/3-540-10000-8_6.

Saxena, A. (2011). "Are Circular Shaped Masks Adequate in Adaptive Mask Overlay Topology Synthesis Method?" *Journal of Mechanical Design* 133.1, 011001. doi: 10.1115/1.4002973.

Schevenels, M.; Lazarov, B. and Sigmund, O. (2011). "Robust Topology Optimization Accounting for Spatially Varying Manufacturing Errors". *Computer Methods in Applied Mechanics and Engineering* 200.49-52, 3613–3627. doi: 10.1016/j.cma.2011.08.006.

Schmidt, R. and Ratto, M. (2013). "Design-To-Fabricate: Maker Hardware Requires Maker Software". *IEEE Computer Graphics and Applications* 33.6, 26–34. doi: 10.1109/mcg.2013.90.

Schmidt, R. and Umetani, N. (2014). "Branching Support Structures for 3D Printing". *ACM SIGGRAPH 2014 Studio on - SIGGRAPH '14*. Association for Computing Machinery (ACM). doi: 10.1145/2619195.2656293.

Schmidt, S. and Schulz, V. (2011). "A 2589 Line Topology Optimization Code Written for the Graphics Card". *Computing and Visualization in Science* 14.6, 249–256. doi: 10.1007/s00791-012-0180-1.

Schuëller, G. I. and Jensen, H. A. (2008). "Computational Methods in Optimization Considering Uncertainties – an Overview". *Computer Methods in Applied Mechanics and Engineering* 198.1, 2–13. doi: 10.1016/j.cma.2008.05.004.

Schüller, C.; Panozzo, D.; GrundhÃśfer, A.; Zimmer, H.; Sorkine, E. and Sorkine-Hornung, O. (2016). "Computational Thermoforming". *ACM Transactions on Graphics* 35.4, 1–9. doi: 10.1145/2897824.2925914.

Schüller, C.; Panozzo, D. and Sorkine-Hornung, O. (2014). "Appearance-Mimicking Surfaces". *ACM Transactions on Graphics* 33.6, 1–10. doi: 10.1145/2661229.2661267.

Schumacher, C.; Bickel, B.; Rys, J.; Marschner, S.; Daraio, C. and Gross, M. (2015). "Microstructures to Control Elasticity in 3D Printing". *ACM Transactions on Graphics* 34.4, 136:1–136:13. doi: 10.1145/2766926.

Schumacher, C.; Thomaszewski, B.; Coros, S.; Martin, S.; Sumner, R. and Gross, M. (2012). "Efficient Simulation of Example-Based Materials". *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 1–8. eprint: https://graphics.ethz.ch/~bthomasz/PDF/ESEBM.pdf.

Schumacher, C.; Thomaszewski, B. and Gross, M. (2016). "Stenciling: Designing Structurally-Sound Surfaces With Decorative Patterns". *Computer Graphics Forum*. doi: 10.1111/cgf.12967.

Schwartzburg, Y. and Pauly, M. (2011). "Design and Optimization of Orthogonally Intersecting Planar Surfaces". *Computational Design Modelling*. Springer Science + Business Media, 191–199. doi: 10.1007/978-3-642-23435-4_22.

Schwartzburg, Y. and Pauly, M. (2013). "Fabrication-Aware Design With Intersecting Planar Pieces". *Computer Graphics Forum* 32.2pt3, 317–326. doi: 10.1111/cgf.12051.

Segerman, H. (2009). *Surface Autoglyphs*. http://www.segerman.org/autologlyphs.html.

Sethian, J. A. and Wiegmann, A. (2000). "Structural Boundary Design via Level Set and Immersed Interface Methods". *Journal of Computational Physics* 163.2, 489–528. doi: 10.1006/jcph.2000.6581.

Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Vol. 3. Cambridge university press.

Shade, J.; Gortler, S.; He, L. and Szeliski, R. (1998). "Layered Depth Images". *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*. Association for Computing Machinery (ACM). doi: 10.1145/280814.280882.

Shin, H. V.; Porst, C. F.; Vouga, E.; Ochsendorf, J. and Durand, F. (2016). "Reconciling Elastic and Equilibrium Methods for Static Analysis". *ACM Transactions on Graphics* 35.2, 1–16. doi: 10.1145/2835173.

Shu, Y.; Qian, Y.; Sun, H. and Chen, Y. (2014). "Efficient Texture Synthesis of Aggregate Solid Material". *The Visual Computer* 30.6-8, 877–887. DOI: 10.1007/s00371-014-0951-4.

Shugrina, M.; Shamir, A. and Matusik, W. (2015). "Fab Forms: Customizable Objects for Fabrication With Validity and Geometry Caching". *ACM Transactions on Graphics* 34.4, 100:1–100:12. DOI: 10.1145/2766994.

Sigmund, O. (2000). "A New Class of Extremal Composites". *Journal of the Mechanics and Physics of Solids* 48.2, 397–428. DOI: 10.1016/s0022-5096(99)00034-4.

Sigmund, O. (2001). "A 99 Line Topology Optimization Code Written in Matlab". *Structural and Multidisciplinary Optimization* 21.2, 120–127. DOI: 10.1007/s001580050176.

Sigmund, O. and Petersson, J. (1998). "Numerical Instabilities in Topology Optimization: A Survey on Procedures Dealing With Checkerboards, Mesh-Dependencies and Local Minima". *Structural Optimization* 16.1, 68–75. DOI: 10.1007/bf01214002.

Sigmund, O. (1994a). "Design of Materials Structures Using Topology Optimization". Doctoral dissertation.

Sigmund, O. (1994b). "Materials With Prescribed Constitutive Parameters: An Inverse Homogenization Problem". *Int. J. Solids Struct.* 31.17, 2313–2329.

Sigmund, O. (1995). "Tailoring Materials With Prescribed Elastic Properties". *Mechanics of Materials* 20.4, 351–368. DOI: 10.1016/0167-6636(94)00069-7.

Sigmund, O. (1997). "On the Design of Compliant Mechanisms Using Topology Optimization". *Mechanics of Structures and Machines* 25.4, 493–524. DOI: 10.1080/08905459708945415.

Sigmund, O. (2007). "Morphology-Based Black and White Filters for Topology Optimization". *Structural and Multidisciplinary Optimization* 33.4-5, 401–424. DOI: 10.1007/s00158-006-0087-x.

Sigmund, O. (2009a). "Manufacturing Tolerant Topology Optimization". *Acta Mechanica Sinica* 25.2, 227–239. DOI: 10.1007/s10409-009-0240-z.

Sigmund, O. (2009b). "Systematic Design of Metamaterials by Topology Optimization". *IUTAM Symposium on Modelling Nanomaterials and Nanosystems*. Springer Science + Business Media, 151–159. DOI: 10.1007/978-1-4020-9557-3_16.

Sigmund, O.; Aage, N. and Andreassen, E. (2016). "On the (Non-)Optimality of Michell Structures". *Structural and Multidisciplinary Optimization*. DOI: 10.1007/s00158-016-1420-7.

Sigmund, O.; Jensen, J. S.; Stolpe, M.; Aage, N.; Andreasen, C. S. and Lazarov, B. S. (2015). "Topology Optimization - Theory, Methods and Applications". PhD. Course (Technical University of Denmark). http://www.kurser.dtu.dk/41591.aspx?menulanguage=en-gb.

Sigmund, O. and Maute, K. (2012). "Sensitivity Filtering From a Continuum Mechanics Perspective". *Structural and Multidisciplinary Optimization* 46.4, 471–475. DOI: 10.1007/s00158-012-0814-4.

Sigmund, O. and Maute, K. (2013). "Topology Optimization Approaches: A Comparative Review". *Structural and Multidisciplinary Optimization* 48.6, 1031–1055. DOI: 10.1007/s00158-013-0978-6.

Skouras, M.; Coros, S.; Grinspun, E. and Thomaszewski, B. (2015). "Interactive Surface Design With Interlocking Elements". *ACM Transactions on Graphics* 34.6, 1–7. doi: 10.1145/2816795.2818128.

Skouras, M.; Thomaszewski, B.; Bickel, B. and Gross, M. (2012). "Computational Design of Rubber Balloons". *Computer Graphics Forum* 31.2pt4, 835–844. doi: 10.1111/j.1467-8659.2012.03064.x.

Skouras, M.; Thomaszewski, B.; Coros, S.; Bickel, B. and Gross, M. (2013). "Computational Design of Actuated Deformable Characters". *ACM Transactions on Graphics* 32.4, 1. doi: 10.1145/2461912.2461979.

Smith, J.; Hodgins, J.; Oppenheim, I. and Witkin, A. (2002). "Creating Models of Truss Structures With Optimization". *ACM Transactions on Graphics* 21.3. doi: 10.1145/566654.566580.

Sokół, T. (2011). "A 99 Line Code for Discretized Michell Truss Optimization Written in Mathematica". *Structural and Multidisciplinary Optimization* 43.2, 181–190. doi: 10.1007/s00158-010-0557-z.

Soler, C.; Cani, M. and Angelidis, A. (2002). "Hierarchical Pattern Mapping". *ACM Trans. Graph.* 21.3, 673–680. doi: 10.1145/566654.566635.

Song, C.; Zhang, H.; Wang, X.; Han, J. and Wang, H. (2014). "Fast Corotational Simulation for Example-Driven Deformation". *Computers & Graphics* 40, 49–57. doi: 10.1016/j.cag.2014.01.003.

Song, P.; Deng, B.; Wang, Z.; Dong, Z.; Li, W.; Fu, C. and Liu, L. (2016). "CofiFab: Coarse-To-Fine Fabrication of Large 3D Objects". *ACM Transactions on Graphics (SIGGRAPH 2016)* 35.4.

Song, P.; Fu, C. and Cohen-Or, D. (2012). "Recursive Interlocking Puzzles". *ACM Transactions on Graphics* 31.6, 1. doi: 10.1145/2366145.2366147.

Song, P.; Fu, C.; Goswami, P.; Zheng, J.; Mitra, N. J. and Cohen-Or, D. (2013). "Reciprocal Frame Structures Made Easy". *ACM Transactions on Graphics* 32.4, 1. doi: 10.1145/2461912.2461915.

Song, P.; Fu, Z.; Liu, L. and Fu, C. (2015). "Printing 3D Objects With Interlocking Parts". *Computer Aided Geometric Design* 35-36, 137–148. doi: 10.1016/j.cagd.2015.03.020.

Stainko, R. (2005). "An Adaptive Multilevel Approach to the Minimal Compliance Problem in Topology Optimization". *Communications in Numerical Methods in Engineering* 22.2, 109–118. doi: 10.1002/cnm.800.

Stainko, R. (2006). *Advanced Multilevel Techniques to Topology Optimization*. na.

Staten, M. L. (2007). "Why Is Hex Meshing So Hard?" Presentation at Sandia National Laboratories.

Stava, O.; Vanek, J.; Benes, B.; Carr, N. and Měch, R. (2012). "Stress Relief: Improving Structural Strength of 3D Printable Objects". *ACM Transactions on Graphics* 31.4, 1–11. doi: 10.1145/2185520.2185544.

Stolpe, M. and Svanberg, K. (2001). "An Alternative Interpolation Scheme for Minimum Compliance Topology Optimization". *Structural and Multidisciplinary Optimization* 22.2, 116–124. doi: 10.1007/s001580100129.

Stolpe, M. (2015). "Truss Topology Optimization With Discrete Design Variables by Outer Approximation". *Journal of Global Optimization* 61.1, 139–163. doi: 10.1007/s10898-014-0142-x.

Strano, G.; Hao, L.; Everson, R. M. and Evans, K. E. (2013). "A New Approach to the Design and Optimisation of Support Structures in Additive Manufacturing". *The International Journal of Advanced Manufacturing Technology* 66.9-12, 1247–1254. doi: 10.1007/s00170-012-4403-x.

Sundararaghavan, V. (2014). "Reconstruction of Three-Dimensional Anisotropic Microstructures From Two-Dimensional Micrographs Imaged on Orthogonal Planes". *Integrating Materials and Manufacturing Innovation* 3.1. doi: 10.1186/s40192-014-0019-3.

Suresh, K. (2010). "A 199-Line Matlab Code for Pareto-Optimal Tracing in Topology Optimization". *Structural and Multidisciplinary Optimization* 42.5, 665–679. doi: 10.1007/s00158-010-0534-6.

Suresh, K. (2013). "Efficient Generation of Large-Scale Pareto-Optimal Topologies". *Structural and Multidisciplinary Optimization* 47.1, 49–61. doi: 10.1007/s00158-012-0807-3.

Sutherland, I. E. (1964). "Sketchpad a Man-Machine Graphical Communication System". *SIMULATION* 2.5, R–20. doi: 10.1177/003754976400200514.

Svanberg, K. (1987). "The Method of Moving Asymptotes—a New Method for Structural Optimization". *International Journal for Numerical Methods in Engineering* 24.2, 359–373. doi: 10.1002/nme.1620240207.

Svanberg, K. (1995). "A Globally Convergent Version of MMA Without Linesearch". *Proceedings of the first world congress of structural and multidisciplinary optimization*. Vol. 28. Goslar, Germany, 9–16.

Svanberg, K. (2002). "A Class of Globally Convergent Optimization Methods Based on Conservative Convex Separable Approximations". *SIAM Journal on Optimization* 12.2, 555–573. doi: 10.1137/s1052623499362822.

Svanberg, K. (2007). *MMA and GCMMA - Two Methods for Nonlinear Optimization*. Tech. rep. Technical report. eprint: https://people.kth.se/~krille/mmagcmma.pdf.

Tagliasacchi, A.; Delame, T.; Spagnuolo, M.; Amenta, N. and Telea, A. (2016). "3D Skeletons: A State-Of-The-Art Report". *Computer Graphics Forum* 35.2, 573–597. doi: 10.1111/cgf.12865.

Takayama, K.; Okabe, M.; Ijiri, T. and Igarashi, T. (2008). "Lapped Solid Textures: Filling a Model With Anisotropic Textures". *ACM SIGGRAPH 2008 papers on - SIGGRAPH '08*. Association for Computing Machinery (ACM). doi: 10.1145/1399504.1360652.

Talischi, C.; Paulino, G. H.; Pereira, A. and Menezes, I. F. M. (2012). "PolyTop: A Matlab Implementation of a General Topology Optimization Framework Using Unstructured Polygonal Finite Element Meshes". *Structural and Multidisciplinary Optimization* 45.3, 329–357. doi: 10.1007/s00158-011-0696-x.

Telea, A. and Jalba, A. (2011). "Voxel-Based Assessment of Printability of 3D Shapes". *Mathematical Morphology and Its Applications to Image and Signal Processing*.

Springer Science + Business Media, 393–404. DOI: 10.1007/978-3-642-21569-8_34.

Thomaszewski, B.; Coros, S.; Gauge, D.; Megaro, V.; Grinspun, E. and Gross, M. (2014). "Computational Design of Linkage-Based Characters". *ACM Transactions on Graphics* 33.4, 1–9. DOI: 10.1145/2601097.2601143.

Thrimurthulu, K.; Pandey, P. M. and Reddy, N. V. (2004). "Optimum Part Deposition Orientation in Fused Deposition Modeling". *International Journal of Machine Tools and Manufacture* 44.6, 585–594. DOI: 10.1016/j.ijmachtools.2003.12.004.

Tomlin, M. and Meyer, J. (2011). "Topology Optimization of an Additive Layer Manufactured (ALM) Aerospace Part". *Proceeding of the 7th Altair CAE technology conference*, 1–9. eprint: http://www.pfonline.com/cdn/cms/uploadedFiles/Topology-Optimization-of-an-Additive-Layer-Manufactured-Aerospace-Part.pdf.

Tong, X.; Zhang, J.; Liu, L.; Wang, X.; Guo, B. and Shum, H. (2002). "Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces". *Proceedings of the 29th annual conference on Computer graphics and interactive techniques - SIGGRAPH '02*. Association for Computing Machinery (ACM). DOI: 10.1145/566570.566634.

Turk, G. (2001). "Texture Synthesis on Surfaces". *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: ACM, 347–354. DOI: 10.1145/383259.383297.

Turner, D. M. and Kalidindi, S. R. (2016). "Statistical Construction of 3-D Microstructures From 2-D Exemplars Collected on Oblique Sections". *Acta Materialia* 102, 136–148. DOI: 10.1016/j.actamat.2015.09.011.

Ulu, E.; Korkmaz, E.; Yay, K.; Ozdoganlar, O. B. and Kara, L. B. (2015). "Enhancing the Structural Performance of Additively Manufactured Objects Through Build Orientation Optimization". *Journal of Mechanical Design* 137.11, 111410. DOI: 10.1115/1.4030998.

Umetani, N.; Bickel, B. and Matusik, W. (2015). "Computational Tools for 3D Printing". *ACM SIGGRAPH 2015 Courses on - SIGGRAPH '15*. Association for Computing Machinery (ACM). DOI: 10.1145/2776880.2792718.

Umetani, N.; Igarashi, T. and Mitra, N. J. (2012). "Guided Exploration of Physically Valid Shapes for Furniture Design". *ACM Transactions on Graphics* 31.4, 1–11. DOI: 10.1145/2185520.2185582.

Umetani, N.; Kaufman, D. M.; Igarashi, T. and Grinspun, E. (2011). "Sensitive Couture for Interactive Garment Modeling and Editing". *ACM SIGGRAPH 2011 papers on - SIGGRAPH '11*. Association for Computing Machinery (ACM). DOI: 10.1145/1964921.1964985.

Umetani, N.; Koyama, Y.; Schmidt, R. and Igarashi, T. (2014). "Pteromys: Interactive Design and Optimization of Free-Formed Free-Flight Model Airplanes". *ACM Transactions on Graphics* 33.4, 1–10. DOI: 10.1145/2601097.2601129.

Umetani, N.; Mitani, J. and Igarashi, T. (2010). "Designing Custom-Made Metallophone With Concurrent Eigenanalysis". *NIME*. Vol. 10. Citeseer, 26–30. eprint: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.478.1030\&rep=rep1\&type=pdf.

Umetani, N.; Panotopoulou, A.; Schmidt, R. and Whiting, E. (2016). "Printone: Interactive Resonance Simulation for Free-Form Print-Wind Instrument Design". *ACM Transactions on Graphics* 35.6, 1–14. doi: 10.1145/2980179.2980250.

Umetani, N. and Schmidt, R. (2013). "Cross-Sectional Structural Analysis for 3D Printing Optimization". *SIGGRAPH Asia 2013 Technical Briefs on - SA '13*. Association for Computing Machinery (ACM). doi: 10.1145/2542355.2542361.

Vallet, B. and Lévy, B. (2008). "Spectral Geometry Processing With Manifold Harmonics". *Computer Graphics Forum* 27.2, 251–260. doi: 10.1111/j.1467-8659.2008.01122.x.

Van Der Burg, M. W. D.; Shulmeister, V.; Van Der Geissen, E. and Marissen, R. (1997). "On the Linear Elastic Properties of Regular and Random Open-Cell Foam Models". *Journal of Cellular Plastics* 33.1, 31–54.

Van Hook, T. (1986). "Real-Time Shaded NC Milling Display". *Proceedings of the 13th annual conference on Computer graphics and interactive techniques - SIGGRAPH '86*. Association for Computing Machinery (ACM). doi: 10.1145/15922.15887.

Vanegas, C. A.; Aliaga, D. G.; Wonka, P.; Müller, P.; Waddell, P. and Watson, B. (2010). "Modelling the Appearance and Behaviour of Urban Spaces". *Computer Graphics Forum*. Vol. 29. 1. Wiley Online Library, 25–42. eprint: https://pdfs.semanticscholar.org/2fa5/555bd7300b7383e62b489169db3dd460533d.pdf.

Vanek, J.; Galicia, J. A. G. and Benes, B. (2014a). "Clever Support: Efficient Support Structure Generation for Digital Fabrication". *Computer Graphics Forum* 33.5, 117–125. doi: 10.1111/cgf.12437.

Vanek, J.; Galicia, J. A. G.; Benes, B.; Měch, R.; Carr, N.; Stava, O. and Miller, G. S. (2014b). "PackMerger: A 3D Print Volume Optimizer". *Computer Graphics Forum* 33.6, 322–332. doi: 10.1111/cgf.12353.

Vanhoey, K.; Sauvage, B.; Larue, F. and Dischler, J. (2013). "On-The-Fly Multi-Scale Infinite Texturing From Example". *ACM Transactions on Graphics* 32.6, 1–10. doi: 10.1145/2508363.2508383.

Verbart, A.; Langelaar, M. and Keulen, F. van (2016). "Damage Approach: A New Method for Topology Optimization With Local Stress Constraints". *Structural and Multidisciplinary Optimization* 53.5, 1081–1098. doi: 10.1007/s00158-015-1318-9.

Vidimce, K.; Kaspar, A.; Wang, Y. and Matusik, W. (2016). "Foundry: Hierarchical Material Design for Multi-Material Fabrication". *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*. Association for Computing Machinery (ACM). doi: 10.1145/2984511.2984516.

Vidimče, K.; Wang, S.; Ragan-Kelley, J. and Matusik, W. (2013). "OpenFab: A Programmable Pipeline for Multi-Material Fabrication". *ACM Transactions on Graphics* 32.4, 1. doi: 10.1145/2461912.2461993.

Vouga, E.; Höbinger, M.; Wallner, J. and Pottmann, H. (2012). "Design of Self-Supporting Surfaces". *ACM Transactions on Graphics* 31.4, 1–11. doi: 10.1145/2185520.2185583.

Wadbro, E. and Berggren, M. (2009). "Megapixel Topology Optimization on a Graphics Processing Unit". *SIAM Review* 51.4, 707–721. doi: 10.1137/070699822.

Wallin, M. and Ristinmaa, M. (2015). "Topology Optimization Utilizing Inverse Motion Based Form Finding". *Computer Methods in Applied Mechanics and Engineering* 289, 316–331. doi: 10.1016/j.cma.2015.02.015.

Wang, C. C. L. and Chen, Y. (2013). "Thickening Freeform Surfaces for Solid Fabrication". *Rapid Prototyping Journal* 19.6, 395–406. doi: 10.1108/rpj-02-2012-0013.

Wang, C. C. L. and Manocha, D. (2013). "GPU-based Offset Surface Computation Using Point Samples". *Computer-Aided Design* 45.2, 321–330. doi: 10.1016/j.cad.2012.10.015.

Wang, F.; Jensen, J. S. and Sigmund, O. (2012). "High-Performance Slow Light Photonic Crystal Waveguides With Topology Optimized or Circular-Hole Based Material Layouts". *Photonics and Nanostructures - Fundamentals and Applications* 10.4, 378–388. doi: 10.1016/j.photonics.2012.04.004.

Wang, F.; Lazarov, B. S. and Sigmund, O. (2011). "On Projection Methods, Convergence and Robust Formulations in Topology Optimization". *Structural and Multidisciplinary Optimization* 43.6, 767–784. doi: 10.1007/s00158-010-0602-y.

Wang, L. and Whiting, E. (2016). "Buoyancy Optimization for Computational Fabrication". *Computer Graphics Forum* 35.2, 49–58. doi: 10.1111/cgf.12810.

Wang, L.; Zhou, K.; Yu, Y. and Guo, B. (2010). "Vector Solid Textures". *ACM SIGGRAPH 2010 papers on - SIGGRAPH '10*. Association for Computing Machinery (ACM). doi: 10.1145/1833349.1778823.

Wang, M. Y. et al. (2005). *Professor Wang's Research Group*. url: http://ihome.ust.hk/~mywang/Download.html.

Wang, M. Y.; Wang, X. and Guo, D. (2003). "A Level Set Method for Structural Topology Optimization". *Computer Methods in Applied Mechanics and Engineering* 192.1-2, 227–246. doi: 10.1016/s0045-7825(02)00559-5.

Wang, W. M.; Zanni, C. and Kobbelt, L. (2016a). "Improved Surface Quality in 3D Printing by Optimizing the Printing Direction". *Computer Graphics Forum* 35.2, 59–70. doi: 10.1111/cgf.12811.

Wang, W.; Chao, H.; Tong, J.; Yang, Z.; Tong, X.; Li, H.; Liu, X. and Liu, L. (2015). "Saliency-Preserving Slicing Optimization for Effective 3D Printing". *Computer Graphics Forum* 34.6, 148–160. doi: 10.1111/cgf.12527.

Wang, W.; Liu, X. and Liu, L. (2014a). "Upright Orientation of 3D Shapes via Tensor Rank Minimization". *Journal of Mechanical Science and Technology* 28.7, 2469–2477. doi: 10.1007/s12206-014-0604-6.

Wang, W.; Wang, T. Y.; Yang, Z.; Liu, L.; Tong, X.; Tong, W.; Deng, J.; Chen, F. and Liu, X. (2013). "Cost-Effective Printing of 3D Objects With Skin-Frame Structures". *ACM Transactions on Graphics* 32.6, 1–10. doi: 10.1145/2508363.2508382.

Wang, X.; Mei, Y. and Wang, M. (2004). "Incorporating Topological Derivatives Into Level Set Methods for Structural Topology Optimization". *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics (AIAA). doi: 10.2514/6.2004-4564.

Wang, Y.; Luo, Z.; Zhang, N. and Kang, Z. (2014b). "Topological Shape Optimization of Microstructural Metamaterials Using a Level Set Method". *Computational Materials Science* 87, 178–186. doi: 10.1016/j.commatsci.2014.02.006.

Wang, Y.; Luo, Z.; Zhang, X. and Kang, Z. (2014c). "Topological Design of Compliant Smart Structures With Embedded Movable Actuators". *Smart Materials and Structures* 23.4, 045024. DOI: 10.1088/0964-1726/23/4/045024.

Wang, Y.; Zhang, L. and Wang, M. Y. (2016b). "Length Scale Control for Structural Optimization by Level Sets". *Computer Methods in Applied Mechanics and Engineering* 305, 891–909. DOI: 10.1016/j.cma.2016.03.037.

Wei, L. (2002). "Texture Synthesis by Fixed Neighborhood Searching". AAI3038169. Doctoral dissertation. Stanford, CA, USA.

Wei, L.; Lefebvre, S.; Kwatra, V. and Turk, G. (2009). "State of the Art in Example-Based Texture Synthesis". *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 93–117. eprint: https://hal.archives-ouvertes.fr/docs/00/60/68/53/PDF/texture_synthesis_eg09star.pdf.

Wei, L. and Levoy, M. (2000). "Fast Texture Synthesis Using Tree-Structured Vector Quantization". *Proc. of SIGGRAPH 2000*, 479–488.

Wei, L. and Levoy, M. (2001). "Texture Synthesis Over Arbitrary Manifold Surfaces". *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*. Association for Computing Machinery (ACM). DOI: 10.1145/383259.383298.

Weyrich, T.; Peers, P.; Matusik, W. and Rusinkiewicz, S. (2009). "Fabricating Microgeometry for Custom Surface Reflectance". *ACM SIGGRAPH 2009 papers on - SIGGRAPH '09*. Association for Computing Machinery (ACM). DOI: 10.1145/1576246.1531338.

Whiting, E.; Ochsendorf, J. and Durand, F. (2009). "Procedural Modeling of Structurally-Sound Masonry Buildings". *ACM SIGGRAPH Asia 2009 papers on - SIGGRAPH Asia '09*. Association for Computing Machinery (ACM). DOI: 10.1145/1661412.1618458.

Whiting, E.; Shin, H.; Wang, R.; Ochsendorf, J. and Durand, F. (2012). "Structural Optimization of 3D Masonry Buildings". *ACM Transactions on Graphics* 31.6, 1. DOI: 10.1145/2366145.2366178.

Williams, J. and Rossignac, J. (2005). "Mason: Morphological Simplification". *Graphical Models* 67.4, 285–303. DOI: 10.1016/j.gmod.2004.10.001.

Worley, S. (1996). "A Cellular Texture Basis Function". *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*. Association for Computing Machinery (ACM). DOI: 10.1145/237170.237267.

Wu, J.; Dick, C. and Westermann, R. (2016a). "A System for High-Resolution Topology Optimization". *IEEE Transactions on Visualization and Computer Graphics* 22.3, 1195–1208. DOI: 10.1109/tvcg.2015.2502588.

Wu, J.; Kramer, L. and Westermann, R. (2016b). "Shape Interior Modeling and Mass Property Optimization Using Ray-Reps". *Computers & Graphics* 58, 66–72. DOI: 10.1016/j.cag.2016.05.003.

Wu, J.; Wang, C. C.; Zhang, X. and Westermann, R. (2016c). "Self-Supporting Rhombic Infill Structures for Additive Manufacturing". *Computer-Aided Design*. DOI: 10.1016/j.cad.2016.07.006.

Wu, R.; Wang, W. and Yu, Y. (2014). "Optimized Synthesis of Art Patterns and Layered Textures". *IEEE Transactions on Visualization and Computer Graphics* 20.3, 436–446. DOI: 10.1109/tvcg.2013.113.

Xia, L. and Breitkopf, P. (2015). "Design of Materials Using Topology Optimization and Energy-Based Homogenization Approach in Matlab". *Structural and Multidisciplinary Optimization* 52.6, 1229–1241. DOI: 10.1007/s00158-015-1294-0.

Xia, L.; Zhu, J.; Zhang, W. and Breitkopf, P. (2013). "An Implicit Model for the Integrated Optimization of Component Layout and Structure Topology". *Computer Methods in Applied Mechanics and Engineering* 257, 87–102. DOI: 10.1016/j.cma.2013.01.008.

Xie, Y.; Xu, W.; Yang, Y.; Guo, X. and Zhou, K. (2015). "Agile Structural Analysis for Fabrication-Aware Shape Editing". *Computer Aided Geometric Design* 35, 163–179. eprint: https://pdfs.semanticscholar.org/5921/9998691bba31c3697f9c032c0977ab16cc36.pdf.

Xin, S.; Lai, C.; Fu, C.; Wong, T.; He, Y. and Cohen-Or, D. (2011). "Making Burr Puzzles From 3D Models". *ACM SIGGRAPH 2011 papers on - SIGGRAPH '11*. Association for Computing Machinery (ACM). DOI: 10.1145/1964921.1964992.

Xu, H.; Li, Y.; Chen, Y. and Barbič, J. (2015a). "Interactive Material Design Using Model Reduction". *ACM Transactions on Graphics (TOG)* 34.2, 18. eprint: http://www-bcf.usc.edu/~yongchen/Research/materialEditor-tog-accepted.pdf.

Xu, H.; Sin, F.; Zhu, Y. and Barbič, J. (2015b). "Nonlinear Material Design Using Principal Stretches". *ACM Transactions on Graphics* 34.4, 75:1–75:11. DOI: 10.1145/2766917.

Xu, K.; Li, H.; Zhang, H.; Cohen-Or, D.; Xiong, Y. and Cheng, Z. (2010). "Style-Content Separation by Anisotropic Part Scales". *ACM Trans. Graph.* 29.6, 184:1–184:10.

Xu, K.; Zhang, H.; Cohen-Or, D. and Chen, B. (2012). "Fit and Diverse: Set Evolution for Inspiring 3D Shape Galleries". *ACM Trans. Graph.* 31.4, 57:1–57:10.

Xu, W.; Li, W. and Liu, L. (2016). "Skeleton-Sectional Structural Analysis for 3D Printing". *Journal of Computer Science and Technology* 31.3, 439–449. DOI: 10.1007/s11390-016-1638-2.

Yang, N.; Tian, Y. and Zhang, D. (2015). "Novel Real Function Based Method to Construct Heterogeneous Porous Scaffolds and Additive Manufacturing for Use in Medical Engineering". *Medical Engineering & Physics* 37.11, 1037–1046. DOI: 10.1016/j.medengphy.2015.08.006.

Yao, M.; Chen, Z.; Luo, L.; Wang, R. and Wang, H. (2015). "Level-Set-Based Partitioning and Packing Optimization of a Printable Model". *ACM Transactions on Graphics* 34.6, 1–11. DOI: 10.1145/2816795.2818064.

Ying, L.; Hertzmann, A.; Biermann, H. and Zorin, D. (2001). "Texture and Shape Synthesis on Surfaces". English. *Rendering Techniques 2001*. Ed. by Gortler, S. and Myszkowski, K. Eurographics. Springer Vienna, 301–312. DOI: 10.1007/978-3-7091-6242-2_28.

Yoshida, H. et al. (2015). "Architecture-Scale Human-Assisted Additive Manufacturing". *ACM Transactions on Graphics (TOG)* 34.4, 88. eprint: https://pdfs.semanticscholar.org/2c38/aa009ced53f8d65a8f445395077577851c98.pdf.

You, Y. H.; Kou, S. T. and Tan, S. T. (2016). "A New Approach for Irregular Porous Structure Modeling Based on Centroidal Voronoi Tessellation and B-Spline". *Computer-Aided Design and Applications* 13.4, 484–489. DOI: 10.1080/16864360.2015.1131542.

Zehnder, J.; Coros, S. and Thomaszewski, B. (2016). "Designing Structurally-Sound Ornamental Curve Networks". *ACM Transactions on Graphics* 35.4, 1–10. DOI: 10.1145/2897824.2925888.

Zhang, J.; Zhang, W.; Zhu, J. and Xia, L. (2012). "Integrated Layout Design of Multi-Component Systems Using XFEM and Analytical Sensitivity Analysis". *Computer Methods in Applied Mechanics and Engineering* 245-246, 75–89. DOI: 10.1016/j.cma.2012.06.022.

Zhang, J.; Zhou, K.; Velho, L.; Guo, B. and Shum, H. (2003). "Synthesis of Progressively-Variant Textures on Arbitrary Surfaces". *ACM Trans. Graph.* 22.3, 295–302. DOI: 10.1145/882262.882266.

Zhang, W.; Xia, L.; Zhu, J. and Zhang, Q. (2011). "Some Recent Advances in the Integrated Layout Design of Multicomponent Systems". *Journal of Mechanical Design* 133.10, 104503. DOI: 10.1115/1.4005083.

Zhang, W.; Yuan, J.; Zhang, J. and Guo, X. (2016a). "A New Topology Optimization Approach Based on Moving Morphable Components (MMC) and the Ersatz Material Model". *Structural and Multidisciplinary Optimization* 53.6, 1243–1260. DOI: 10.1007/s00158-015-1372-3.

Zhang, W.; Zhong, W. and Guo, X. (2014). "An Explicit Length Scale Control Approach in SIMP-based Topology Optimization". *Computer Methods in Applied Mechanics and Engineering* 282, 71–86. DOI: 10.1016/j.cma.2014.08.027.

Zhang, W.; Zhong, W. and Guo, X. (2015a). "Explicit Layout Control in Optimal Design of Structural Systems With Multiple Embedding Components". *Computer Methods in Applied Mechanics and Engineering* 290, 290–313. DOI: 10.1016/j.cma.2015.03.007.

Zhang, W.; Zheng, J. and Thalmann, N. M. (2015b). "Real-Time Subspace Integration for Example-Based Elastic Material". *Computer Graphics Forum* 34.2, 395–404. DOI: 10.1111/cgf.12569.

Zhang, X.; Xia, Y.; Wang, J.; Yang, Z.; Tu, C. and Wang, W. (2015c). "Medial Axis Tree—an Internal Supporting Structure for 3D Printing". *Computer Aided Geometric Design* 35-36, 149–162. DOI: 10.1016/j.cagd.2015.03.012.

Zhang, X.; Le, X.; Panotopoulou, A.; Whiting, E. and Wang, C. C. (2015d). "Perceptual Models of Preference in 3D Printing Direction". *ACM Transactions on Graphics (TOG)* 34.6, 215. eprint: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.710.582\&rep=rep1\&type=pdf.

Zhang, X.; Le, X.; Wu, Z.; Whiting, E. and Wang, C. C. L. (2016b). "Data-Driven Bending Elasticity Design by Shell Thickness". *Computer Graphics Forum*. DOI: 10.1111/cgf.12972.

Zhang, Y.; Yin, C.; Zheng, C. and Zhou, K. (2015e). "Computational Hydrographic Printing". *ACM Transactions on Graphics* 34.4, 131:1–131:11. DOI: 10.1145/2766932.

Zhao, H.; Hong, C.; Lin, J.; Jin, X. and Xu, W. (2016a). "Make It Swing: Fabricating Personalized Roly-Poly Toys". *Computer Aided Geometric Design* 43, 226–236. doi: 10.1016/j.cagd.2016.02.001.

Zhao, H. et al. (2016b). "Connected Fermat Spirals for Layered Fabrication". *ACM Transactions on Graphics* 35.4, 1–10. doi: 10.1145/2897824.2925958.

Zhou, K.; Huang, X.; Wang, X.; Tong, Y.; Desbrun, M.; Guo, B. and Shum, H. (2006). "Mesh Quilting for Geometric Texture Synthesis". *ACM Transactions on Graphics* 25.3, 690. doi: 10.1145/1141911.1141942.

Zhou, M. and Rozvany, G. I. N. (1991). "The COC Algorithm, Part II: Topological, Geometrical and Generalized Shape Optimization". *Computer Methods in Applied Mechanics and Engineering* 89.1-3, 309–336. doi: 10.1016/0045-7825(91)90046-9.

Zhou, M.; Lazarov, B. S. and Sigmund, O. (2014a). "Topology Optimization for Optical Projection Lithography With Manufacturing Uncertainties". *Applied Optics* 53.12, 2720. doi: 10.1364/ao.53.002720.

Zhou, M.; Lazarov, B. S.; Wang, F. and Sigmund, O. (2015). "Minimum Length Scale in Topology Optimization by Geometric Constraints". *Computer Methods in Applied Mechanics and Engineering* 293, 266–282. doi: 10.1016/j.cma.2015.05.003.

Zhou, M. and Wang, M. Y. (2013). "Engineering Feature Design for Level Set Based Structural Optimization". *Computer-Aided Design* 45.12, 1524–1537. doi: 10.1016/j.cad.2013.06.016.

Zhou, Q. and Jacobson, A. (2016). "Thingi10K: A Dataset of 10, 000 3D-Printing Models". *CoRR* abs/1605.04797.

Zhou, Q.; Panetta, J. and Zorin, D. (2013). "Worst-Case Structural Analysis". *ACM Transactions on Graphics* 32.4, 1. doi: 10.1145/2461912.2461967.

Zhou, S. and Li, Q. (2008). "Design of Graded Two-Phase Microstructures for Tailored Elasticity Gradients". *Journal of Materials Science* 43.15, 5157–5167. doi: 10.1007/s10853-008-2722-y.

Zhou, S.; Jiang, C. and Lefebvre, S. (2014b). "Topology-Constrained Synthesis of Vector Patterns". *ACM Transactions on Graphics* 33.6, 1–11. doi: 10.1145/2661229.2661238.

Zhou, Y.; Zhang, W.; Zhu, J. and Xu, Z. (2016). "Feature-Driven Topology Optimization Method With Signed Distance Function". *Computer Methods in Applied Mechanics and Engineering* 310, 1–32. doi: 10.1016/j.cma.2016.06.027.

Zhu, F.; Li, S. and Wang, G. (2014). "Example-Based Materials in Laplace-Beltrami Shape Space". *Computer Graphics Forum* 34.1, 36–46. doi: 10.1111/cgf.12457.

Zillober, C. (1993). "A Globally Convergent Version of the Method of Moving Asymptotes". *Structural Optimization* 6.3, 166–174. doi: 10.1007/bf01743509.

Zuo, Z. H. and Xie, Y. M. (2015). "A Simple and Compact Python Code for Complex 3D Topology Optimization". *Advances in Engineering Software* 85, 1–11. doi: 10.1016/j.advengsoft.2015.02.006.

Abstract

The main goal of this thesis is to propose methods to synthesize shapes in a controllable manner, with the purpose of being fabricated. As 3D printers grow more accessible than ever, modeling software must now take into account fabrication constraints posed by additive manufacturing technologies. Consequently, efficient algorithms need to be devised to model the complex shapes that can be created through 3D printing. We develop algorithms for by-example shape synthesis that consider the physical behavior of the structure to fabricate. All the contributions of this thesis focus on the problem of generating complex shapes that follow geometric constraints and structural objectives.

In a first time, we focus on dealing with fabrication constraints, and propose a method for synthesizing efficient support structures that are well-suited for filament printers. In a second time, we take into account appearance control, and develop new by-example synthesis methods that mixes in a meaningful manner criteria on the appearance of the synthesized shapes, and constraints on their mechanical behavior. Finally, we present a highly scalable method to control the elastic properties of printed structures. We draw inspiration from procedural texture synthesis methods, and propose an efficient algorithm to synthesize printable microstructures with controlled elastic properties.

Keywords: 3D Printing, Fabrication Constraints, Shape Synthesis, Modeling, By-Example Synthesis, Procedural Texturing, Topology Optimization.

Résumé

L'objet principal de cette thèse est de proposer des méthodes pour la synthèse de formes qui soient contrôlables et permettent d'imprimer les résultats obtenus. Les imprimantes 3D étant désormais plus faciles d'accès que jamais, les logiciels de modélisation doivent maintenant prendre en compte les contraintes de fabrication imposées par les technologies de fabrication additives. En conséquence, des algorithmes efficaces doivent être développés afin de modéliser les formes complexes qui peuvent être créées par impression 3D. Nous développons des algorithmes pour la synthèse de formes par l'exemple qui prennent en compte le comportement mécanique des structures devant être fabriquées. Toutes les contributions de cette thèse s'intéressent au problème de génération de formes complexes sous contraintes géométriques et objectifs structurels.

Dans un premier temps, nous nous intéressons à la gestion des contraintes de fabrication, et proposons une méthode pour synthétiser des structures de support efficaces qui sont bien adaptées aux imprimantes à filament. Dans un deuxième temps, nous prenons en compte le contrôle de l'apparence, et développons de nouvelles méthodes pour la synthèse par l'exemple qui mélangent astucieusement des critères sur visuels, et des contraintes sur le comportement mécanique des objets. Pour finir, nous présentons une méthode passant bien à l'échelle, afin de contrôler les propriétés élastiques des structures imprimées. Nous nous inspirons des méthodes de synthèse de texture procédurales, et proposons un algorithme efficace pour synthétiser des microstructures imprimables et contrôler leurs propriétés élastiques.

Mots-clefs : Impression 3D, Contraintes de fabrication, Synthèse de formes, Modélisation, Synthèse par l'exemple, Texturation procédural, Optimisation topologique.