

**Université de Montréal**

**Stability-Aware Simplification of Curve Networks**

par

**William Neveu**

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique

juillet 2022



# Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

## **Stability-Aware Simplification of Curve Networks**

présenté par

**William Neveu**

a été évalué par un jury composé des personnes suivantes :

*Margarida Carvalho*

---

(présidente-rapporteuse)

*Mikhail Bessmeltsev*

---

(directeur de recherche)

*Pierre Poulin*

---

(membre du jury)



# Résumé

---

La conception de réseaux de courbes nécessite la considération de plusieurs facteurs: la stabilité de la structure, l'efficacité matérielle, et l'aspect esthétique — des objectifs complexes et interdépendants rendant la conception manuelle difficile.

Nous présentons une nouvelle méthode permettant de simplifier des réseaux de courbes destinés à la fabrication. Pour un ensemble de courbes 3D donné, notre algorithme en sélectionne un sous-ensemble stable. Bien que la stabilité soit traditionnellement mesurée par l'ordre de grandeur des déformations entraînées par des charges prédéfinies, une telle approche peut s'avérer limitante. Elle ne tient ni compte des effets de vibration pour les structures de grandes tailles, ni des multiples possibilités de forces appliquées pour les structures et objets de plus petite taille. Ainsi, nous optimisons directement pour une déformation minimale avec la charge dans le pire des cas (de l'anglais *worst-case*).

Notre contribution technique est une nouvelle formulation de la simplification de réseaux de courbes pour la stabilité dans le pire des cas. Celle-ci mène à un problème d'optimisation semi-définie positive en nombres entiers (MI-SDP). Malgré que résoudre ce problème MI-SDP directement est irréaliste dans la plupart des cas, une intuition physique nous mène à un algorithme vorace efficace. Enfin, nous démontrons le potentiel de notre approche à l'aide plusieurs réseaux de courbes et validons l'efficacité de notre méthode en la comparant de façon quantitative à des approches plus simples.

**Mots clés:** réseaux de courbes, stabilité, design pour fabrication



# Abstract

---

Designing curve networks for fabrication requires simultaneous consideration of structural stability, cost effectiveness, and visual appeal — complex, interrelated objectives that make manual design a difficult and tedious task. We present a novel method for fabrication-aware simplification of curve networks, algorithmically selecting a stable subset of given 3D curves. While traditionally, stability is measured as the magnitude of deformation induced by a set of predefined loads, predicting applied forces for common day objects can be challenging. Instead, we directly optimize for minimal deformation under the worst-case load.

Our technical contribution is a novel formulation of 3D curve network simplification for worst-case stability, leading to a mixed-integer semi-definite programming problem (MI-SDP). We show that while solving MI-SDP directly is impractical, a physical insight suggests an efficient greedy heuristic algorithm. We demonstrate the potential of our approach on a variety of curve network designs and validate its effectiveness compared to simpler alternatives using numerical experiments.

**Keywords:** curve networks, stability, fabrication-aware design





# Contents

---

<b>Résumé</b> .....	5
<b>Abstract</b> .....	7
<b>List of Tables</b> .....	13
<b>List of Figures</b> .....	15
<b>List of Acronyms and Abbreviations</b> .....	17
<b>Acknowledgements</b> .....	19
<b>Chapter 1. Introduction</b> .....	21
Architecture and Engineering .....	21
Free-form Architecture .....	22
Curve Networks .....	24
<b>Chapter 2. Related Work</b> .....	27
2.1. Fabrication-Aware Surface Design .....	27
2.2. Stability Optimization .....	28
2.3. Eigenvalue Optimization .....	29
2.4. Computational Design of Curve Networks .....	31
2.5. Truss Topology Optimization for Structural Design .....	32

<b>Chapter 3. Methodology</b> .....	35
3.1. Curves.....	35
3.2. Intersections .....	37
3.3. Elastic Energy .....	37
3.4. Optimization .....	38
3.5. Validation .....	39
<b>Chapter 4. Stability-Aware Simplifaction of Curve Networks</b> .....	41
Contributions.....	44
4.1. Introduction .....	45
4.2. Related Work.....	47
4.2.1. Fabrication-Aware Surface Design .....	47
4.2.2. Stability Optimization .....	48
4.2.3. Eigenvalue Optimization .....	48
4.2.4. Computational Design of Curve Networks .....	49
4.2.5. Truss Topology Optimization For Structural Design .....	49
4.3. Curve Network Simplification .....	51
4.3.1. Computational Model of a Curve Network .....	51
4.3.2. Simplification Framework .....	53
4.3.3. Mixed-Integer Semi-definite Programming (MISDP) Formulation .....	56
4.4. Solver Mechanism.....	58
4.5. Results, Validation, and Discussion.....	59
<b>Chapter 5. Conclusion</b> .....	65

References ..... 67



## List of Tables

---

3.1	Curve tracing approaches per example.....	36
4.1	Algorithm statistics for different curve networks.....	62



# List of Figures

---

1.1	Classical examples of architectural curved surfaces .....	22
1.2	Contemporary examples of architectural free-form surfaces .....	23
1.3	Close-up of the panels at the Walt Disney Concert Hall .....	23
1.4	Deployment of the Toledo gridshell 2.0 .....	25
1.5	Examples of the irregular curve network aesthetic .....	26
2.1	Compression vs tension .....	28
2.2	Gravitational vs worst-case load .....	28
3.1	Generating a geodesic curve .....	36
3.2	Representation of fixed intersections .....	37
3.3	Sampled surfaces for comparison .....	39
4.1	Example of input and output curve networks .....	45
4.2	Examples of curve networks .....	47
4.3	Truss topology optimization VS curve network optimization .....	50
4.4	Curve representation .....	52
4.5	Influence of parameter $\alpha_i$ .....	54
4.6	Nullspace of a curve segments .....	54
4.7	Iterative results .....	56
4.8	Relaxation of the binary constraint .....	56

4.9	Solving for equilibrium.....	60
4.10	Comparing our algorithm to a brute force approach.....	60
4.11	Comparing results: random algorithm and our algorithm.....	61
4.12	A gallery of additional results.....	62



## List of Acronyms and Abbreviations

---

3D	Three-dimensional
MI	Mixed integer
SDP	Semi-definite programming
PSD	Positive semi-definite
TTO	Truss topology optimization



## Acknowledgements

---

I would like to thank my family and friends for their presence and support.

A special thank to Ivan, for welcoming me to the lab and co-authoring two papers with me.

To Ulric for his help demystifying too many compilation errors.

To Robin for letting me use his printer.

And to Misha for his guidance and patience.



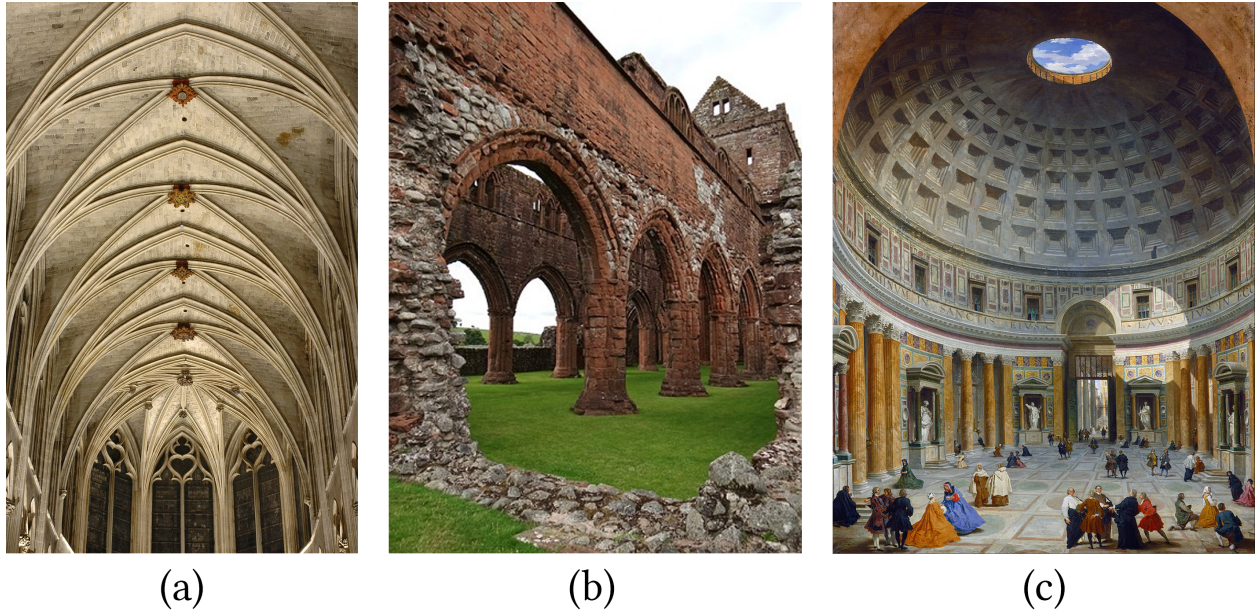
# Chapter 1

---

## Introduction

### Architecture and Engineering

Architecture is part art form, part engineering. While the overall goal — to create a building or another structure is the same, each part has its own challenge. As an art, architecture focuses on the visual appearance as well as intrinsic aspect of a project: circulation, lighting, space use, etc. Staying true to the architect’s vision is paramount. From the engineering standpoint, the focus is structural stability, i.e., constructing buildings that do not break down at the first gust of wind. Stability concerns will always trump design intents when making final decisions, so a designer needs to be aware of stability constraints early in the design process. Failure to do so often shows up as ad-hoc extra structural elements (think of columns in the middle of a hallway), or can lead to drastic changes in the design, steering the project away from the architect’s original intent. For many architects, designing while taking stability factors into account is challenging, since they are not trained engineers. Even for architects with some engineering training, thinking about those aspects may distract from the creative process. In our work we aim to help architects in their design process by creating intuitive tools that take stability constraints into account.



**Fig. 1.1.** Classical examples of architectural curved surfaces. (a) The rib vault ceiling of the Saint-Séverin church in France. (b) Arches at the Sweetheart Abbey in Scotland. (c) Painting of the Pantheon’s dome in Italy. Images by Roman Bonnefoy, [designingbuildings.co.uk](http://designingbuildings.co.uk), Giovanni Paolo Panini respectively.

## Free-form Architecture

If traditional architecture can be described as straight lines, planes, and right angles, free-form architecture is all about smooth curved surfaces. Curved surfaces have been used at least since antiquity, often as vaults or domes (Fig. 1.1). Modern free-form architecture, however, leverages the power of computers and 3D modeling software to create complex and often irregular curved surfaces. The past few decades of the digital age have introduced such complex creations as the works of Zaha Hadid (Fig. 1.2a) and Frank Gehry (Fig. 1.2b).

Despite its visual appeal, free-form architecture remains rare. One reason might be that the process of capturing an architect’s vision in a digital surface ready for fabrication is difficult. This process is a complex interplay of three components: 3D surface modeling, rationalization, and physical simulation. First, during 3D modeling, the architect digitally renders their vision of the surface 3D shape with the help of modeling tools. Second, the digital surface has to be approximated via smaller individual parts that will form it when assembled, a process called *rationalization*. For Figure 1.3, this involves designing each panel,



(a)



(b)

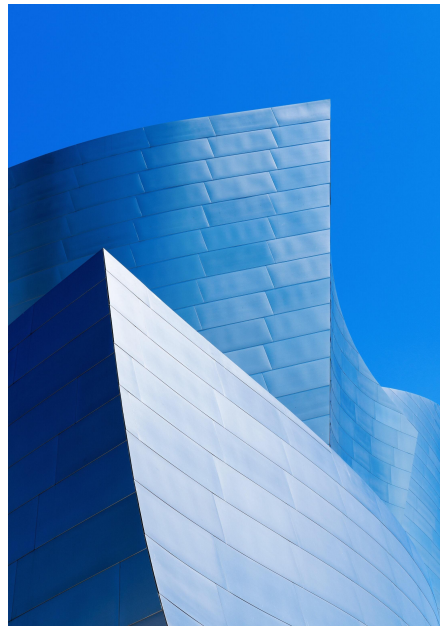


(c)



(d)

**Fig. 1.2.** Contemporary examples of architectural free-form surfaces. (a) Heydar Aliyev Centre in Baku, Azerbaijan, designed by Zaha Hadid Architects. (b) Walt Disney Concert Hall in Los Angeles, USA, designed by Frank Gehry. (c) The Great Court in London, England, designed by Foster + Partners. (d) Yas Hotel in Abu Dhabi, UAE, designed by ARUP. Images by Andrea Pistolessi, Sharad Raval, Nigel Young, ARUP respectively.



**Fig. 1.3.** Close-up of the rectangular panels at the Walt Disney Concert Hall. Image by spacecadet-01 on reddit.com.

with some of them potentially being unique in dimension or curvature. Third, a digital structure undergoes a physical simulation and analysis to evaluate its stability. The stability issues discovered during simulation are addressed in the next design iteration. Hopefully, after a few iterations, the structure is ready for fabrication.

As discussed previously, the issues discovered during the simulation and analysis stage often may lead to undesired changes that go against the designer’s intent. While this is also true for traditional architecture, the problem is exacerbated for free-form architecture. Having more freedom makes it hard to find references, i.e., similar realized projects, to learn from. Without references on structural stability, a large part of the structure’s design becomes guess work. In this thesis, we develop an intuitive design tool that takes into account rationalization methods and structural stability.

## Curve Networks

One interesting rationalization of free-form structures is the curve network (Figs. 1.4, 1.5). A curve network is built from multiple rods bent into the shape of a target surface. A popular class of curve networks is the gridshell (Fig. 1.4). They use long flexible rods intersecting each other regularly in a gridlike fashion. They are attractive for industrial applications, because the rods can all be assembled on the ground (Fig. 1.4a) and subsequently deployed by placing their ends in predetermined positions resulting in a curved surface (Fig. 1.4b). However, gridshells come with a downside: they are limited to regular grid patterns. Irregular curve networks, as their name implies, allow for less “tidy” patterns (Fig. 1.5). These pattern can be attractive, but they remain rare because there are no software tools allowing for easy design.

We set out to develop a tool to help design irregular curve networks. We would like a user to be able to provide curves they believe are meaningful to their design — for their aesthetic, density, placement, etc. — but not necessarily stable. Our tool should automatically add and remove curves to maximize the structure’s stability, while staying true to the designer’s intents.





**Fig. 1.4.** Deployment of the Toledo gridshell 2.0. The gridshell is first assembled on a flat surface (a), before being deployed (b,c). The fully deployed gridshell (d) stands on its own (scaffolding was later removed). This figure is taken directly from [D’Amico et al., 2015].

This brings us to the following problem statement.

**Problem statement:** Given a set of curves that intersect each other on a surface, select a subset that maximizes the structure’s worst-case stability, while respecting a provided material budget.

As this is a master’s thesis by article, we will first explore how our problem fits in related work. Then, we briefly talk about methodology and explain specifics of the problem we will be solving in the article. After that, we present the article forming the core of this thesis, with no modification except for formatting. We end this thesis with a short discussion on possible improvements and future work.



(a)



(b)



(c)

**Fig. 1.5.** Irregular curve networks used in architecture. This unique aesthetic is rarely seen at such scales. (a) Beijing National Stadium in Beijing, China, designed by Herzog & de Meuron, ARUP, and China Architecture Design & Research Group. (b) Hans Wilsdorf bridge in Geneva, Switzerland, by Brodbeck-Roulet SA. (c) Serpentine Gallery Pavilion 2002 in London, England, by Toyo Ito and Cecil Balmond. Images by [www.china.org.cn](http://www.china.org.cn), Cimolai, Sylvain Deleu respectively.

# Chapter 2

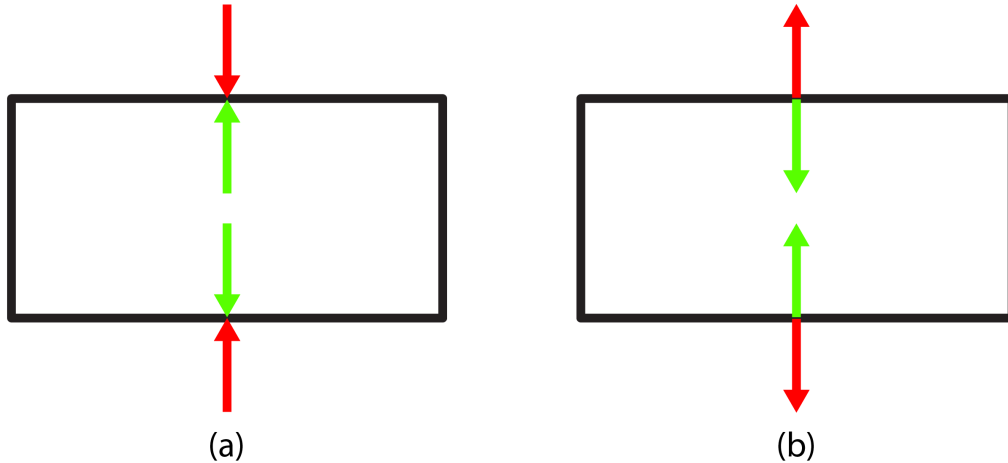
---

## Related Work

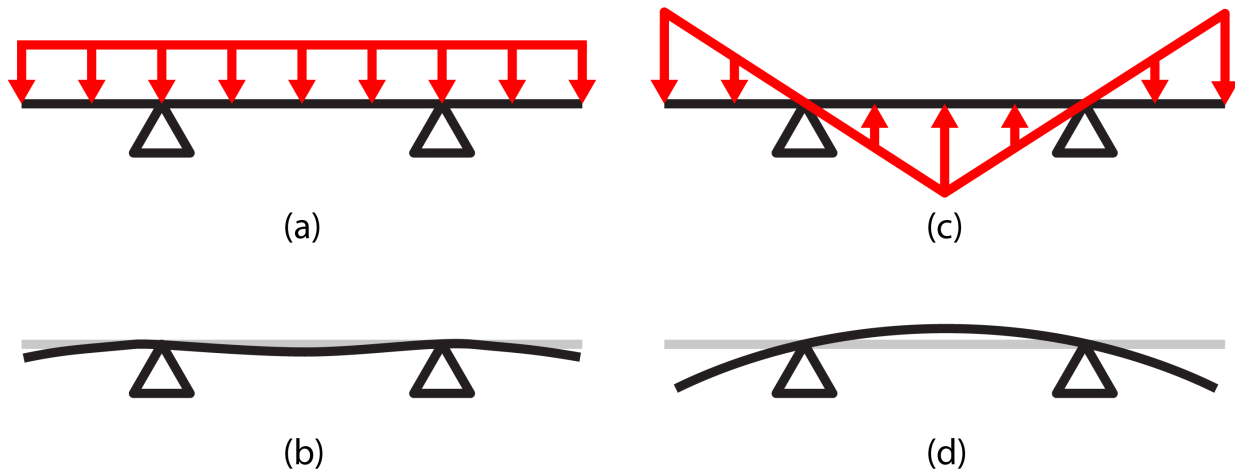
Here, while trying to situate our work into a body of publications, we will try to ease you into the problem, its setup and some of our decisions. This chapter is mostly the same as the *Related Work* section of the article presented in this thesis (Sec. 4.3). However, you will find some additional explanations and publications related to our own. We will present related work in fabrication-aware surface design, stability optimization, eigenvalue optimization, computational design of curve networks, and truss topology optimization for structural design.

### 2.1. Fabrication-Aware Surface Design

As explained in the introduction chapter (Chap. 1), building a surface from an assembly of curves is an example of what we call rationalization. Generally, the fabrication of large surfaces requires rationalization, as building them as one big piece is impractical or impossible. Other fabrication-aware rationalization approaches include self-supporting surfaces [Liu et al., 2013; Panozzo et al., 2013; Vouga et al., 2012; de Goes et al., 2013], auxetics [Konaković-Luković et al., 2018], zippables [Schüller et al., 2018], and others. As their name indicates, self-supporting surfaces work to mitigate their own weight. For example, Panozzo et al. [2013] rationalize the surface with bricks. Bricks below carry the weight of bricks above. This is called a compression-only structure (Fig. 2.1). Every element, in this case a brick, is compressed but never tensioned. In contrast, we consider both compression and tension.



**Fig. 2.1.** We consider an element, in this case a brick, under compression when outside forces (red) are aimed towards its interior (a). When the outside forces are aimed away from its interior, the brick is under tension (b). In either case, the element will cancel out outside forces (green), if it is within its capabilities.



**Fig. 2.2.** A majority of loads of a fixed norm (a) result in small deformations (b). However, one can find a load of the same norm (c), which maximizes the deformation of the bar (d). Note that there is a second worst-case load in this situation: the horizontal reflection of the load in (c).

## 2.2. Stability Optimization

Our main focus when adding or removing curves from a network is that it remains stable. Detecting and reducing structural weaknesses in objects designed for 3D printing is a problem that has received considerable attention from the graphics community. One line of research aims at identifying regions of high stresses using simulation with user-defined or heuristically

determined loads [Stava et al., 2012; Lu et al., 2014]. Instead of depending on pre-defined applied forces, Langlois et al. [2016] use stochastic optimization to estimate worst-case loads. Cui et al. [2020] expand on this approach with a linear-time algorithm for probability gradient computation. Rather than relying on purely stochastic forces, Schumacher et al. [2018] account for uncertainties in load locations and directions by parameterizing the space of expected deviations. While they optimize for worst-case loads within a low-dimensional subspace, we use an eigenvalue optimization method that considers all possible load directions simultaneously. The idea of using eigenanalysis to discover structural weaknesses has been explored before, including work by de Gournay et al. [2008], Zhou et al. [2013], Zehnder et al. [2016]. These approaches use eigenanalysis to detect weak regions, but resort to other means for improving strength. In contrast, our method directly maximizes its worst-case stability criterion, i.e., the minimum eigenvalue of the system’s stiffness matrix. As shown in Figure 2.2, worst-case leads to larger deformations than widely used gravitational loads. It is important to note that whether the worst-case load appears natural or not is irrelevant. This is due to the fact that the worst-case load is related to the worst vibration mode of the structure. It is our prerogative that we should always consider such vibration modes if we are to guarantee stability.

### 2.3. Eigenvalue Optimization

Our worst-case stability criterion gives rise to a constrained optimization problem with bounds on the minimum eigenvalue of a matrix. First, let us review the physical intuition that allows us to use eigenanalysis to measure stability. Given a single spring – elastic system where Hooke’s law applies:  $\vec{F} = k \vec{x}$ , with force vector  $\vec{F}$ , stiffness coefficient  $k$ , and displacement vector  $\vec{x}$ . This also stands in the case of a system of multiple springs, but  $k$  will be replaced by a symmetric positive semi-definite stiffness matrix  $H$ . Therefore, we have:

$$\vec{F} = H \vec{x},$$

which models the relationship between force and displacement in the system. Next, we define the worst-case load as the force of fixed norm that induces the displacement of greatest magnitude, or inversely, as the force of smallest norm that induces a displacement of fixed norm. For a given structure, we can find the worst-case load and displacement *pair* by solving the following optimization model:

$$\begin{aligned} \min_{\vec{x}} \quad & \|H \vec{x}\| \\ \text{s.t.} \quad & \|\vec{x}\| = 1. \end{aligned}$$

In turn, since  $H$  is a symmetric PSD matrix, the previous model is equivalent to this second optimization model:

$$\begin{aligned} \min_{\vec{x}} \quad & \frac{1}{2} \vec{x}^T H \vec{x} \\ \text{s.t.} \quad & \|\vec{x}\|^2 = 1. \end{aligned}$$

Now, this model can easily be expressed through a Lagrangian polynomial:

$$L = \frac{1}{2} \vec{x}^T H \vec{x} - \lambda (\vec{x}^T \vec{x} - 1).$$

By solving

$$\nabla L = H \vec{x} - \lambda \vec{x} = 0,$$

we can finally get  $\vec{x}$ . However, we also see from the last equation that this problem is the eigensystem  $H \vec{x} = \lambda \vec{x}$ . This indicates that the worst-case load will be an eigenvector of  $H$ . By rearranging  $H \vec{x} = \vec{F} = \lambda \vec{x}$ , we get:

$$\lambda = \frac{|\vec{F}|}{|\vec{x}|}.$$

From this, we can clearly see that the smaller the norm of the force vector  $\vec{F}$  needs to be to generate a displacement vector  $\vec{x}$  of a given norm. This means that the worst-case load will in fact be the eigenvector of  $\lambda_1$ , the *smallest* eigenvalue of  $H$ . Consequently, we define the stability of the structure as  $\lambda_1$ , and will try to maximize this value.

Eigenvalue optimization problems occur naturally in many applications of engineering design, e.g., when tuning the frequency response of a structure [Torigaki et al., 1994] or optimizing a continuous elastic structure to minimize its worst-case compliance [Cherkaev and Cherkaev, 2004]. In the graphics community, eigenvalue optimization problems have been investigated, e.g., in geometry processing as a means of enforcing bounds on deformation [Kovalsky et al., 2014]. Our approach likewise gives rise to a semi-definite programming problem, but whereas those works optimize over a set of continuous parameters, our decision variables are binary.

Eigenvalue optimization problems have also been investigated in the context of computational design using both gradient-free [Bharaj et al., 2015] and gradient-based methods [Musialski et al., 2016; Panetta et al., 2017]. Rather than optimizing for target eigenvalues, Chen et al. [2017] aim at preserving the smallest eigenvalue during mesh coarsening for elastodynamics applications. A generalization of this idea to spectrum-preserving coarsening of geometric linear operators was proposed by Liu et al. [2019]. While all of these methods work on continuous variables, we address the discrete problem of selecting an optimal subset from a large set of predefined candidate curves.

## 2.4. Computational Design of Curve Networks

A number of work consider the design of curve networks on surfaces. One line of research focuses on biaxial or triaxial weaving to create regular ribbon structures [Vekhter et al., 2019; Campen and Kobbelt, 2014; Akleman et al., 2009; Takezawa et al., 2016; Tao et al., 2016, 2017; Ren et al., 2021]. Other physical curve networks that have been explored include wire meshes [Garg et al., 2014], structures made from planar pre-bent rods [Miguel et al., 2016] and circular arcs [Bo et al., 2011], 3D-printed curve networks [Pérez et al., 2015; Pérez et al., 2017], and regular gridshells [Pillwein and Musialski, 2021; Panetta et al., 2019; Lienhard and Knippers, 2015; Schling et al., 2018]. While the above methods focus on highly regular networks and/or fixed topology, we target curve networks that are not necessarily regular with *a priori* unknown combinatorics.

We are inspired by the work of Zehnder et al. [2016] where they tile a surface with a repeated curve pattern, optimizing for connectedness and low deformation of the network. However, they only identify weak areas in their network. Instead, we directly optimize the topology of our curve network for maximal stability. Following previous work on curve network design, our method relies on discrete elastic rods [Bergou et al., 2008, 2010] and its extension to network connections [Pérez et al., 2015; Panetta et al., 2019] as an underlying simulation model.

## 2.5. Truss Topology Optimization for Structural Design

Our optimization method (Sec. 4.3) is inspired by progress in truss topology optimization (TTO) for structural systems, where a network of straight *bars* connected at a predefined set of *nodes* is optimized to sustain a fixed load or satisfy a vibration constraint (Fig. 4.3). We outline only the most relevant work, see [Stolpe, 2016] for an in-depth overview.

Many TTO approaches, starting from the classical work of Dorn et al. [1964], find thicknesses of the straight bars, assuming a predefined set of nodes where the loads are applied. For instance, [Ben-Tal and Nemirovski, 1997] optimize straight bar widths with respect to a set of load cases. Achtziger and Kočvara [2007] introduce TTO formulations using the minimum eigenvalues to analyze the free vibrations of the structures. More recently, Kočvara [2015] considers a variant of TTO, where widths can take only integer values, including 0, leading to an integer linear SDP problem. These work rely on knowing beforehand the set of nodes where the bars connect, necessary to define loads or vibration modes. In our case, however, nodes, i.e., curve intersections, are not fixed and may appear or disappear during optimization, leading to a strictly harder problem.

A small number of work address this problem by including the set of nodes as binary variables in the overall optimization, focusing on constraining feasible topologies [Cerveira et al., 2010] or preventing buckling [Mela, 2014].



In a recent work, Arora et al. [2019] find a regular, structurally sound truss configuration given a load, using mesh parameterization and quad meshing methods, where parameterization directions are aligned with principal stresses. Jiang et al. [2019] optimize structure weight and compliance, given a load case, via a combination of geometrical and topological optimization.



# Chapter 3

---

## Methodology

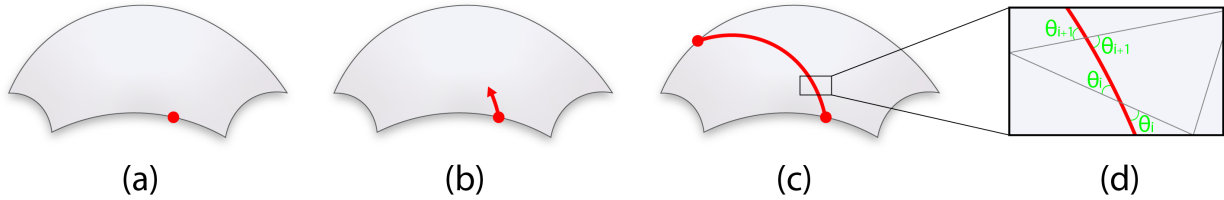
In order to solve our problem, we need to make choices as to which approaches we use to address every facet of the problem. In this chapter, we will flesh out our objective and provide explanations on our choice of approaches.

### 3.1. Curves

As indicated in the problem statement (Chap. 1), a curve network needs to be generated prior to solving the problem. There are multiple ways to create a curve network. The first solution is to trace the curves manually, drawing them in a 3D modeling software. In our case, we do so with Rhinoceros 3D [McNeel et al., 2010]. The curves then need to be imported by the algorithm. The second solution is to algorithmically trace the curves. Most often we do so by drawing geodesics on the surface (Fig. 3.1). Otherwise, we trace the curves by guiding them with a frame field — an assignment of two directions per point, defined over the surface. We noticed that input curve networks geared toward equal density over the target surface generally result in prettier results. As this is more easily achieved with manually drawn curves, we did not limit ourselves to only using algorithmically generated curves (Table 3.1).

Curves themselves are modeled as polylines. This representation provides multiple advantages:

- Allows curves to be traced over meshes.



**Fig. 3.1.** The process used to generate the curves we describe as *geodesics*. (a) Choose a point on the surface, preferably on a boundary. (b) Choose a *shooting* direction for the curve. (c,d) Trace a polyline along the mesh and stop once a boundary of the surface is hit.

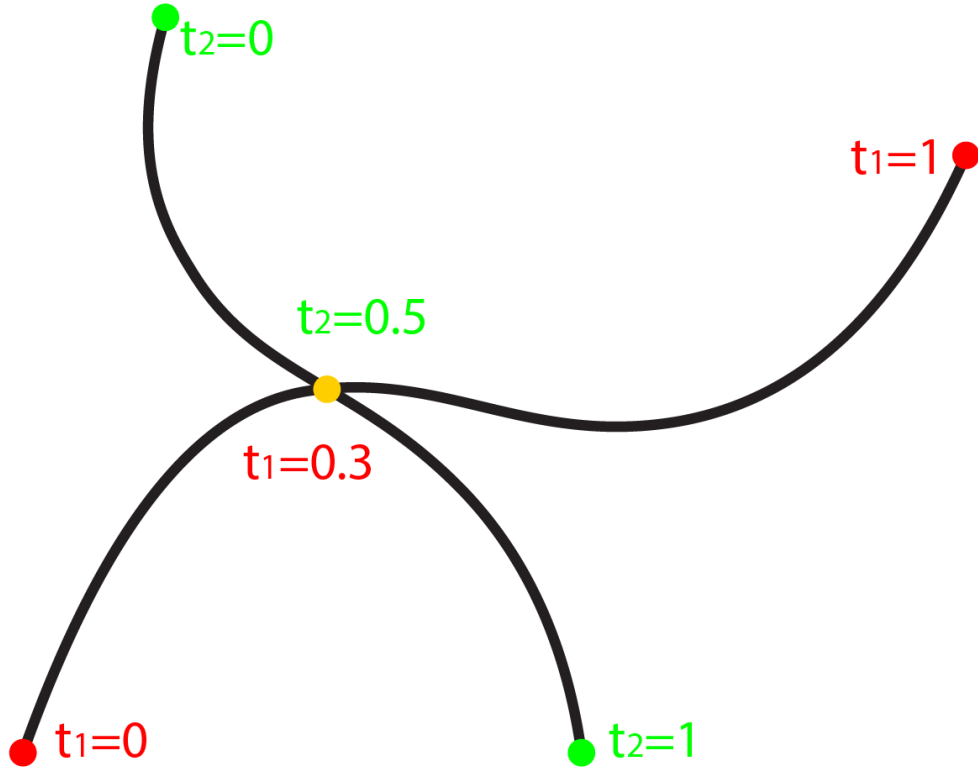
Surface	Fig.	# curves generated by		
		Geodesics	Frame field	Manually drawn
hill	4.1	70	0	3
kagome	4.7a	0	0	114
roof	4.7b	0	0	75
stadium	4.11a	100	0	0
tower	4.11d	0	0	33
shell	4.12a	0	63	0
bunny	4.12b	75	0	0
tent	4.12c	76	0	36
arched shell	4.12d	0	63	7

**Table 3.1.** Curve tracing approaches per example.

- Easy to read and manipulate.
- Shares many similarities to trusses (Fig. 4.3).
- Can be used in conjunction with discrete elastic energies [Bergou et al., 2008, 2010].

We simulate every curve to be a bent straight rod with isotropic bending response. This greatly simplifies our equations and implementation. However, the discrete elastic rods' model of Bergou et al. [2008, 2010] also covers elastic energies for pre-bent rods and anisotropic bending response. The radius and stiffness of the rods can be specified by the user. Nonetheless, for practicality, every example in the article used the same rod properties.

The final user input to the algorithm is the material budget. It gives us an upper bound for the total length of the curves in the output of the algorithm. During optimization, this upper bound will be used to constrain the system. In practice, this parameter controls the density of the resulting curve network.



**Fig. 3.2.** When two curves intersect, they do so at a certain point (yellow) along their length (red and green). As long as these two curves are present in a curve network, their intersection will always happen at curve parameters  $t_1 = 0.3$  and  $t_2 = 0.5$ , prohibiting them to *slide* along each other. The intersection can still move in 3D space.

## 3.2. Intersections

By the nature of our problem, curves must intersect so that they work together to become more stable. As curves are all traced over a common surface, they naturally cross each other. In practice, we model intersections as shared vertices between the intersecting curves. We allow intersections to move in space, but not to move along their intersecting curves (Fig. 3.2). Lastly, our elastic energy (Sec. 4.3) allows us to simply sum up per-curve energy at an intersection vertex to obtain the energy of that vertex.

## 3.3. Elastic Energy

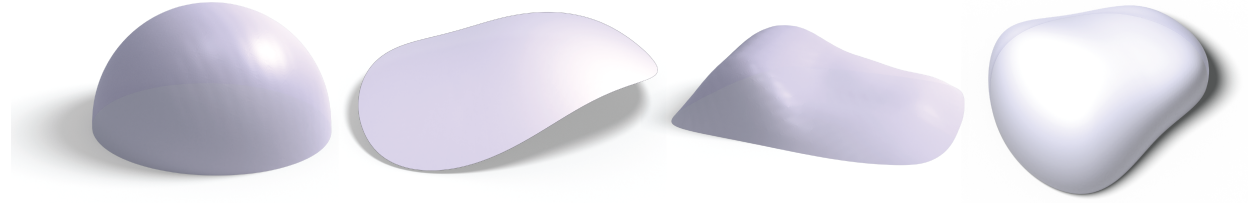
The energy of an elastic rod should normally be defined as the sum of its bending, stretching and twisting energies [Bergou et al., 2008, 2010]. We deliberately chose to not

rely on twist as a load-bearing mechanism. Following common practice in truss design, we do not enforce relative orientations between curves at intersections since this would lead to large moments at the joints. Furthermore, we neither constrain cross-sectional orientations of curves at intersections, nor at their ends. Since all curves are initially straight and their cross-sectional orientations are not constrained, the lowest energy configuration will always be twist-free, regardless of how bent curves are [Bergou et al., 2008]. Consequently, cross-sectional orientations and twist are not needed in our model.

### 3.4. Optimization

We optimize the curve networks under the worst-case load instead of user prescribed, and possibly more intuitive load cases. Knowing which loads to brace against is a central challenge in structural engineering. Even if the probability distribution of uncertain loads is known, one must brace for the worst-case if critical failure is a risk. We argue that, unless worst-case loads can safely be excluded, ignoring them for architectural design would be misguided. The worst-case loads we consider have a clear physical interpretation: the eigenvector corresponding to the minimum Hessian eigenvalue is directly connected to the lowest-frequency vibration mode of the structure.

Since we are selecting a subset of curves, it is evident that our problem is a binary assignment one: from an initial set, we must select a subset minimizing some energy. Either the curves from the initial set are kept or they are removed. An alternative approach would have been to make our variables associated with the presence of each curve continuous, for example, controlling each curve’s radius. But, such an approach makes fabrication harder. To fabricate such a network, one must have access to rods of multiple and possibly unique diameters. We opt for a simpler fabrication process by assuming that every curve in the network has the same properties.



**Fig. 3.3.** The surfaces used to trace the random curve networks used in Figure 4.10.

### 3.5. Validation

As we solve a novel problem, there is no pre-existing algorithms for us to compare to. However, we devised two simple approaches to compare our work to. First, a brute-force algorithm that computes the stability of every feasible subset for a given set of curves. The complexity of this algorithm is  $\mathcal{O}(2^n)$ . Therefore, we only use it for small initial curve networks. In practice, computing the brute-force solution for a single network of 20 curves takes multiple hours. We compare smallest eigenvalues, as well as the percentile of our solution among all feasible combinations. Second, a random search algorithm: we time our own algorithm, then use this as a time limit for an algorithm that evaluates the stability of random feasible subsets. This approach serves as a baseline, which we aim to outperform. The curve networks used to gather data for Figure 4.10 are random geodesics traced over four simple surfaces (Fig. 3.3).





## Chapter 4

# Stability-Aware Simplification of Curve Networks

William Neveu

This article was published in ACM Transactions on Graphics  
(SIGGRAPH 2022)



# Stability-Aware Simplification of Curve Networks

by

William Neveu<sup>1</sup>, Ivan Puhachov<sup>1</sup>, Bernhard Thomaszewski<sup>2</sup>, and Mikhail Bessmeltsev<sup>1</sup>

(<sup>1</sup>) Université de Montréal

(<sup>2</sup>) ETH Zürich

This article was published in ACM Transactions on Graphics (SIGGRAPH 2022).

## Contributions

As first author, my contributions are various. I have participated in the elaboration of both algorithms, implemented them, and produced the data for all figures. Furthermore, I wrote the first draft of the article and was actively involved in further modifications leading to the final manuscript.

Ivan Puhachov is a fellow student under the supervision of Mikhail Bessmeltsev. Prior to this publication, we had worked together on his SIGGRAPH article on 2-dimensional sketch vectorization [Puhachov et al., 2021]. For the article that follows, he produced Figures 4.1, 4.4, 4.5, 4.7, 4.9, 4.11, and 4.12. He also implemented an algorithm that traces frame field-guided curves on meshes. Due to time constraints, only Figures 4.12a and 4.12d used curves generated by this tool.

With the most experience in the field of digital fabrication and physics-based simulation, Prof. Bernhard Thomaszewski gave us unique insights that we would otherwise have been missing. As for the writing, he participated to the abstract, the introduction, and the related work sections. Lastly, his help was instrumental when writing a rebuttal to the SIGGRAPH reviewers, which ultimately got this paper accepted for publication.

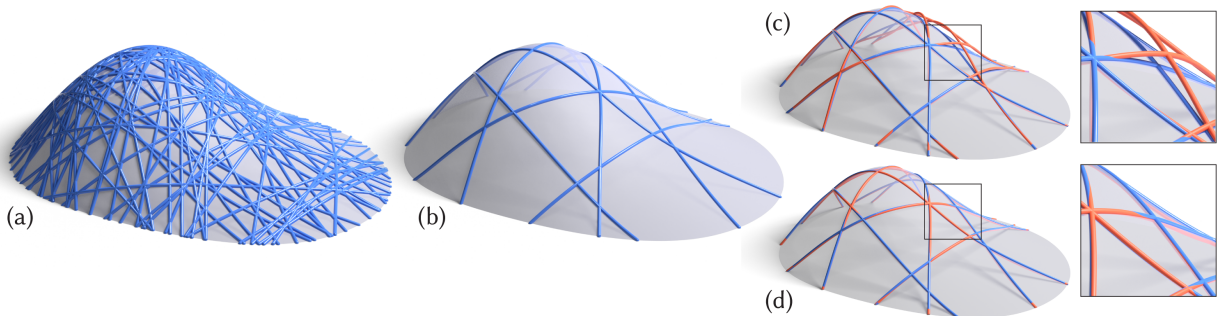
As my supervisor, Prof. Mikhail Bessmeltsev helped shape and guide the article to its final state. From suggestions on experimentation to crucial ideas such as the use of SDP constraints, this article would not exist without his help. He made sure that I did my fair share of thinking and lifting, always making our discussions back-and-forths where my ideas were welcome.

ABSTRACT. *This abstract is identical to the one presented on page 7.*

Designing curve networks for fabrication requires simultaneous consideration of structural stability, cost effectiveness, and visual appeal — complex, interrelated objectives that make manual design a difficult and tedious task. We present a novel method for fabrication-aware simplification of curve networks, algorithmically selecting a stable subset of given 3D curves. While traditionally, stability is measured as the magnitude of deformation induced by a set of predefined loads, predicting applied forces for common day objects can be challenging. Instead, we directly optimize for minimal deformation under the worst-case load.

Our technical contribution is a novel formulation of 3D curve network simplification for worst-case stability, leading to a mixed-integer semi-definite programming problem (MI-SDP). We show that while solving MI-SDP directly is impractical, a physical insight suggests an efficient greedy heuristic algorithm. We demonstrate the potential of our approach on a variety of curve network designs and validate its effectiveness compared to simpler alternatives using numerical experiments.

**Keywords:** curve networks, stability, fabrication-aware design



**Fig. 4.1.** We automatically simplify a user-provided curve network on a surface (a) to fit a given material budget with nearly optimal stability (b). Since the directions of applied loads are hard to predict, we optimize for *worst-case* stability and minimize the maximum deformation induced by a worst-case external force of unit norm in total (c), implying smaller deformations for all other unit-norm forces, in particular, for the standard gravitational load (d).

## 4.1. Introduction

From birds’ nests to modern stadiums, from weaved baskets to environmental sculpture, from spiderwebs to rooftops, curve networks are found commonly in nature, used in art and crafts, architecture and engineering (Fig. 4.2). Made out of wicker, wire, bent wooden or

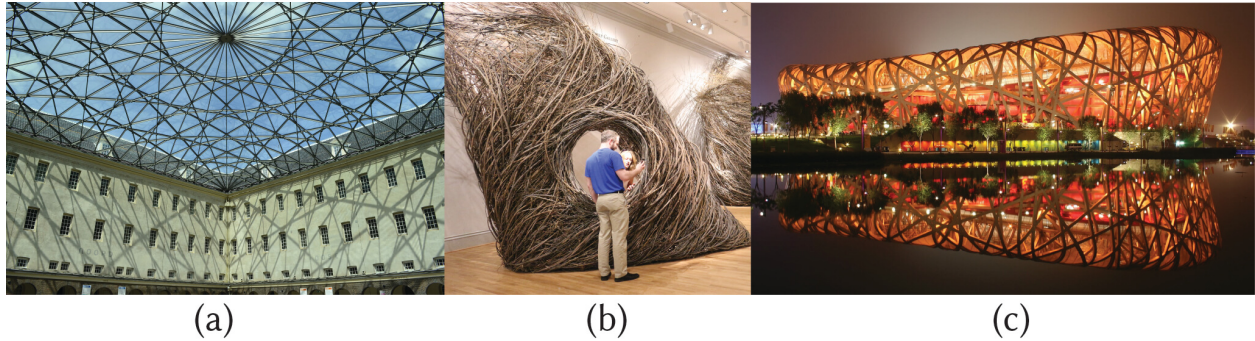
metallic beams, these curves are an aesthetically pleasing and functional means to convey complex 3D surfaces. However, designing such curve networks for fabrication is challenging. Artists have to manage requirements of very different, often contradicting, nature: create appealing designs, use little material, and make sure the structure is stable. This often forces artists to follow a cumbersome process where they iteratively adjust their design and test its stability via simulation or prototyping.

In traditional drawing, artists often ideate via a freeform rough sketch, which they then clean up, taking engineering requirements into account [Eissen and Steur, 2007]. Similarly, one natural workflow to create a 3D curve network is to first draft a complex curve network on the target surface, guided only by aesthetics, then simplify it, taking into account engineering and budget constraints.

Inspired by this metaphor, we present a new method for fabrication-aware simplification of a given curve network. More specifically, our method selects a subset of given curves on a 3D surface to maximize the curve network’s *worst-case* stability. The initial set of curves may be created by sketching interfaces [Arora and Singh, 2020], traditional modeling software [Autodesk, 2022], or various curve tracing algorithms. We demonstrate that our method produces complex yet stable simplified curve networks (Fig. 4.1).

Traditionally, stability of a structure is expressed as the amount of deformation induced by a particular set of loads. Yet, loads for common-day objects, small-scale architecture, or sculptures are hard to predict. Therefore, we focus on an alternative definition of stability, measuring the deformation of the structure under the *worst-case* load, i.e., an external force vector of fixed magnitude but *a priori* of unknown direction. Thus, our problem is selecting a subset of the given curves, within a fixed material budget, maximizing the structure’s worst-case stability.

Our technical innovation is a novel formulation of this problem, where the notion of worst-case stability, related to eigenanalysis of the network’s stiffness matrix, leads us to a mixed-integer semi-definite programming (MI-SDP) instance. MI-SDP problems are known to be NP-hard and we show that optimization via modern solvers is infeasible. Instead,



**Fig. 4.2.** Examples of curve networks in architecture, visual art, and design. (a) The roof of the National Maritime Museum in Amsterdam. (b) A sculpture by Patrick Dougherty at Smithsonian American Art Museum. (c) Beijing National Stadium. Images by BristollIcarus, Eb0178a (cropped), Peter23 respectively, licensed under CC BY-SA.

we leverage intuition from the physical world that suggests an elegant and simple greedy approximation approach, which we demonstrate to be efficient and accurate.

We validate our method numerically by analyzing its approximation properties, comparing it to a random subset selection scheme, and demonstrating its applications on a gallery of curve networks.

## 4.2. Related Work

Our work is inspired by the progress in two areas: computational design of curve networks and truss topology optimization. We focus only on the most relevant work.

### 4.2.1. Fabrication-Aware Surface Design

The graphics community has seen a significant progress in fabrication-aware design of surfaces, including self-supporting surfaces [Liu et al., 2013; Panozzo et al., 2013; Vouga et al., 2012; de Goes et al., 2013], auxetic [Konaković-Luković et al., 2018], zippables [Schüller et al., 2018], and others. These work often focus on either surface approximation only [Schüller et al., 2018] or surface-specific aspects such as paneling and mold reuse [Eigensatz et al., 2010; Fu et al., 2010; Pellis et al., 2021] or face regularity [Vaxman et al., 2017]. In contrast, we focus on simplification of curve networks, whose faces may not be regular or even closed (Fig. 4.1).

### 4.2.2. Stability Optimization

Detecting and improving structural weaknesses in objects designed for 3D printing is a problem that has received considerable attention from the graphics community. One line of research aims at identifying regions of high stresses using simulation with user-defined or heuristically determined loads [Stava et al., 2012; Lu et al., 2014]. Instead of depending on predefined applied forces, Langlois et al. [2016] use stochastic optimization to estimate worst-case loads. Cui et al. [2020] expand on this approach with a linear-time algorithm for probability gradient computation. Rather than relying on purely stochastic forces, Schumacher et al. [2018] account for uncertainties in load locations and directions by parameterizing the space of expected deviations. While they optimize for worst-case loads within a low-dimensional subspace, our eigenvalue optimization method considers all possible load directions simultaneously. The idea of using eigenanalysis to discover structural weaknesses has been explored before, including work by de Gournay et al. [2008], Zhou et al. [2013], Zehnder et al. [2016]. These approaches use eigenanalysis to detect weak regions, but resort to other means for improving strength. In contrast, our method directly maximizes its worst-case stability measure, i.e., the minimum eigenvalue of the system’s stiffness matrix.

### 4.2.3. Eigenvalue Optimization

Our worst-case stability criterion gives rise to a constrained optimization problem with bounds on the minimum eigenvalue of the stiffness matrix. Eigenvalue optimization problems occur naturally in many applications of engineering design, e.g., when tuning the frequency response of a structure [Torigaki et al., 1994] or optimizing a continuous elastic structure to minimize its worst-case compliance [Cherkaev and Cherkaev, 2004]. In the graphics community, eigenvalue optimization problems have been investigated, e.g., in geometry processing as a means of enforcing bounds on deformations [Kovalsky et al., 2014]. Our approach likewise gives rise to a semi-definite programming problem, but whereas those work optimize over a set of continuous parameters, our decision variables are binary. Eigenvalue optimization problems have also been investigated in the context of computational design using



both gradient-free [Bharaj et al., 2015] and gradient-based methods [Musialski et al., 2016; Panetta et al., 2017]. Rather than optimizing for target eigenvalues, Chen et al. [2017] aim at preserving the smallest eigenvalue during mesh coarsening for elastodynamics applications. A generalization of this idea to spectrum-preserving coarsening of geometric linear operators was proposed by Liu et al. [2019]. While all of these methods work on continuous variables, we address the discrete problem of selecting an optimal subset from a large set of predefined candidate curves.

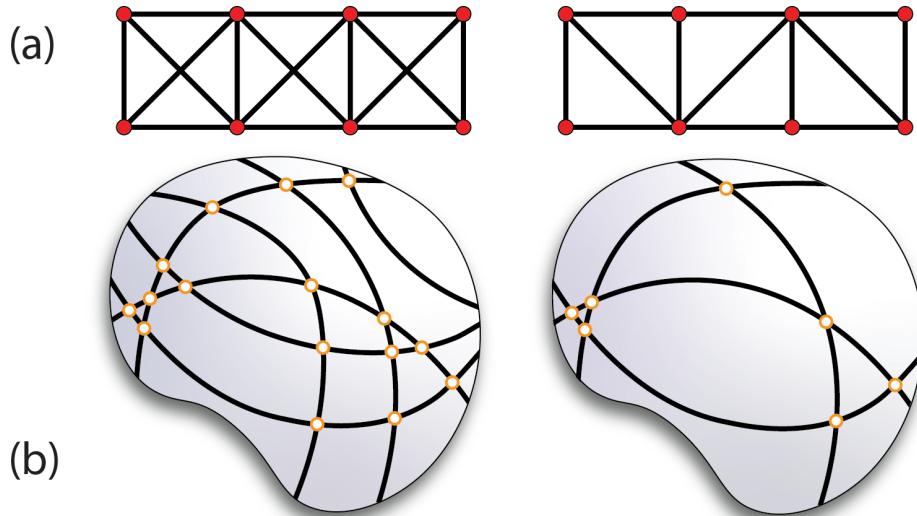
#### 4.2.4. Computational Design of Curve Networks

A number of work consider the design of curve networks on surfaces. One line of research focuses on biaxial or triaxial weaving to create regular ribbon structures [Vekhter et al., 2019; Campen and Kobbelt, 2014; Akleman et al., 2009; Takezawa et al., 2016; Tao et al., 2016, 2017; Ren et al., 2021]. Other physical curve networks that have been explored include wire meshes [Garg et al., 2014], structures made from planar pre-bent rods [Miguel et al., 2016] and circular arcs [Bo et al., 2011], 3D-printed curve networks [Pérez et al., 2015; Pérez et al., 2017], and regular gridshells [Pillwein and Musialski, 2021; Panetta et al., 2019; Lienhard and Knippers, 2015; Schling et al., 2018]. While the above methods focus on highly regular networks and/or fixed topology, we target curve networks that are not necessarily regular with *a priori* unknown combinatorics.

Following previous work on curve network design, our method relies on discrete elastic rods [Bergou et al., 2008, 2010] and its extension to network connections [Pérez et al., 2015; Panetta et al., 2019] as an underlying simulation model.

#### 4.2.5. Truss Topology Optimization For Structural Design

Our optimization method (Sec. 4.3) is inspired by progress in truss topology optimization (TTO) for structural systems, where a network of straight *bars* connected at a predefined set of *nodes* is optimized to sustain a fixed load or satisfy a vibration constraint (Fig. 4.3). We outline only the most relevant work; see [Stolpe, 2016] for an in-depth overview.



**Fig. 4.3.** Typical truss topology optimization approaches (a) optimize a 2D/3D network of straight bars (black) connected at predefined nodes (red), either optimizing the compliance for a set of loads or satisfying vibration constraints. In contrast, we simplify a curve network on a given surface where the set of nodes, i.e., curve intersections, is not known beforehand (b).

Many TTO approaches, starting from the classical work of Dorn et al. [1964], find thicknesses of the straight bars, assuming a predefined set of nodes where the loads are applied. For instance, Ben-Tal and Nemirovski [1997] optimize straight bar widths with respect to a set of load cases. Achtziger and Kočvara [2007] introduce TTO formulations using the minimum eigenvalues to analyze the free vibrations of the structure. More recently, Kočvara [2015] considers a variant of TTO, where widths can take only integer values, including zero, leading to an integer linear SDP problem. These work rely on knowing beforehand the set of nodes where the bars connect, necessary to define loads or vibration modes. In our case, however, nodes, i.e., curve intersections, are not fixed and may appear or disappear during optimization, leading to a strictly harder problem.

A small number of work address this problem by including the set of nodes as binary variables in the overall optimization, focusing on constraining feasible topologies [Cerveira et al., 2010] or preventing buckling [Mela, 2014].

In a recent work, Arora et al. [2019] find a regular, structurally sound truss configuration given a load, using mesh parameterization and quad meshing methods, where parameterization directions are aligned with principal stresses. Jiang et al. [2019] optimize structure weight and compliance, given a load case, via a combination of geometrical and topological optimizations.

Trusses, however, support the load only via *compression* of *straight* rods. In our setup, however, both *compression and bending* of *curvilinear* rods are important load-bearing mechanisms, requiring a different formulation.

### 4.3. Curve Network Simplification

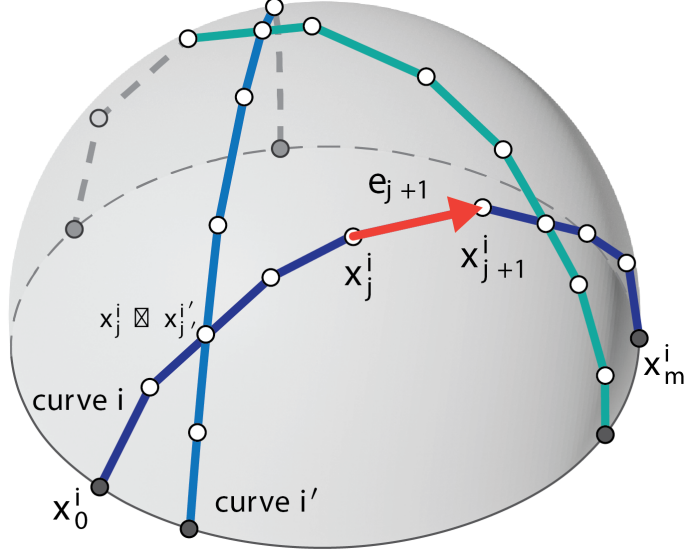
At the foundation of our method lies the mathematical model to simulate the deformation of a network of bent rods. We first focus on the physical model for the simulation of a curve network with known geometry and connectivity (Sec. 4.3.1), then describe our algorithm to simplify a curve network (Sec. 4.3.2).

#### 4.3.1. Computational Model of a Curve Network

We model each curve, a polyline initially lying on the target mesh, as a discrete elastic rod [Bergou et al., 2008, 2010]. Curves are connected to each other at the intersection points and their endpoints are fixed at the boundary of the given surface (Fig. 4.4).

For simplicity of exposition, we assume each rod is straight when manufactured and then bent during assembly. Initial curvature can be incorporated into the model with no changes to our algorithm. We do not enforce relative curve orientations at intersections since this would lead to large moments at the joints. Furthermore, we neither constrain cross sectional orientations of curves at intersections, nor at their endpoints. The lowest energy configuration thus will always be twist-free, regardless of how bent curves are. We therefore omit the twisting energy from Bergou et al. [2008].

Therefore, the elastic energy of each curve  $i$  of  $m + 1$  vertices with vertex coordinates  $x_j^i \in \mathbb{R}^3, j = 0, \dots, m$  is a sum of stretching and bending terms



**Fig. 4.4.** Curves are represented as polylines on the input mesh. Intersecting curves share their common vertices. Each curve is modeled as a discrete elastic rod.

$$E_{\text{stretch}}^i = \sum_{j=1}^m E_{\text{stretch}}^{i,j} = \sum_{j=1}^m k_s \left( \frac{|e_j|}{|\bar{e}_j|} - 1 \right)^2 |\bar{e}_j| \quad (4.3.1)$$

$$E_{\text{bend}}^i = \sum_{j=1}^{m-1} E_{\text{bend}}^{i,j} = \sum_{j=1}^{m-1} \frac{k_b}{\ell_j} \|\kappa_j\|_2^2, \quad (4.3.2)$$

with the binormal curvature

$$\kappa_j = \frac{2e_{j-1} \times e_j}{\|\bar{e}_{j-1}\| \cdot \|\bar{e}_j\| + e_{j-1} \cdot e_j},$$

where  $k_b$  and  $k_s$  are the rods' bending and stretching stiffness coefficients respectively,  $\bar{e}_j$  is an edge of the original polyline that becomes  $e_j = x_j^i - x_{j-1}^i$  after the deformation, and  $\ell_j = 0.5(\|\bar{e}_j\| + \|\bar{e}_{j-1}\|)$ . Here we omit some superscripts  $i$  for brevity. Note that the stretching energy is discretized per edge, while bending energy depends on an angle between edges and hence is discretized per vertex.

Each pair of intersecting curves  $i, i'$  is coupled via a shared vertex, i.e.,  $x_j^i \equiv x_j^{i'}$ . Physically, this is equivalent to a rotational joint at the intersection allowing curve rotations but not translations. The first and the last vertices of each curve are fixed via a soft penalty

constraint with  $w = 80$ :

$$E_{\text{endpoints}}^i = w(\|x_0^i - \bar{x}_0^i\|^2 + \|x_m^i - \bar{x}_m^i\|^2). \quad (4.3.3)$$

The total energy of the curve network with  $n$  curves is then a sum of their the stretching, bending, and endpoint energies:

$$E = \sum_{i=1}^n E_{\text{stretch}}^i + E_{\text{bend}}^i + E_{\text{endpoints}}^i. \quad (4.3.4)$$

### 4.3.2. Simplification Framework

Using this basic model, we now formulate our simplification problem. This stage chooses a subset of the curve network within a given budget, maximizing the worst-case stability of the structure. Here we define material budget as an upper bound on the total length of the curves in the simplified network.

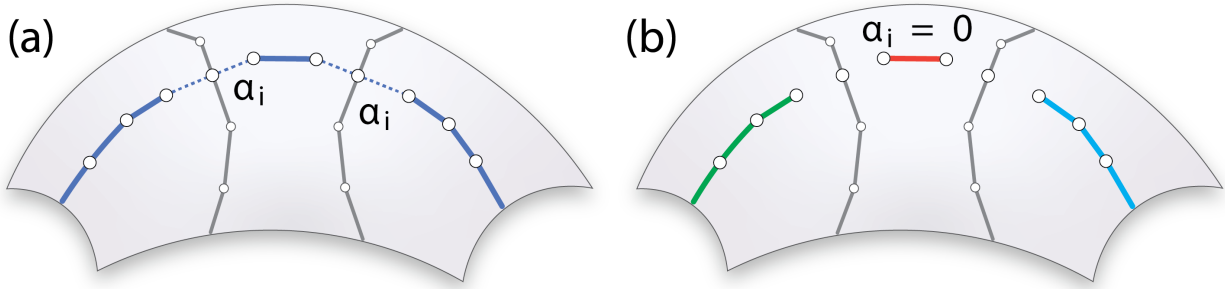
For each curve  $i = 1, \dots, n$ , we need to decide whether to keep it or reject it, encoded as  $\alpha_i = 1$  or  $\alpha_i = 0$ , respectively. In the following algorithm, we perform a relaxation  $\alpha_i \in [0, 1]$ , so we treat  $\alpha_i$  as a factor controlling stiffnesses  $k_s, k_b$  of two edges around each intersection of that curve with the other curves (Fig. 4.5a). Thus,  $\alpha_i$  can be seen as a factor of cross-section area of those edges for the stretching energy and of the squared area for the bending energy.

To define this formally, let  $\mathcal{V}$  be the set of the vertices shared between curve  $i$  and the curves intersecting it. Then we can rewrite energies Eqs. 4.3.1 and 4.3.2 as functions of  $\alpha_i$  (Fig. 4.5):

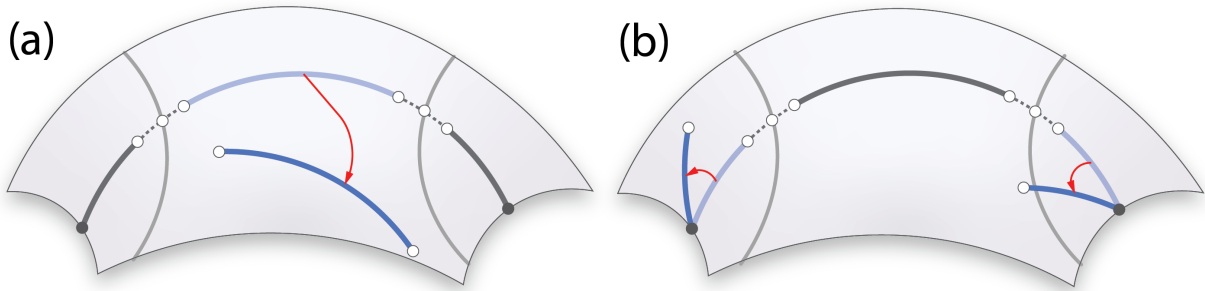
$$E_{\text{stretch}}^i(\alpha_i) = \sum_{j \text{ and } j+1 \notin \mathcal{V}} E_{\text{stretch}}^{i,j} + \alpha_i \sum_{j \text{ or } j+1 \in \mathcal{V}} E_{\text{stretch}}^{i,j} \quad (4.3.5)$$

$$E_{\text{bend}}^i(\alpha_i) = \sum_{j \notin \mathcal{V}} E_{\text{bend}}^{i,j} + \alpha_i \sum_{j \in \mathcal{V}} E_{\text{bend}}^{i,j}. \quad (4.3.6)$$

Then, grouping the terms with and without  $\alpha_i$  and denoting the corresponding partial sums as  $E^i$  and  $E_0^i$  respectively, the total energy in Eq. 4.3.4 can then be expressed as:



**Fig. 4.5.** For each curve, the parameter  $\alpha_i$  controls the stiffness of the two edges next to the intersections (dashed lines) with other curves (a). We interpret the extreme case of  $\alpha_i = 0$ , which effectively splits the curve into disconnected pieces, as removing the curve from the network.



**Fig. 4.6.** Once a curve is disconnected from the rest of the network, some segments can be easily translated and rotated (a) and some can be rotated around the fixed endpoint (b).

$$E(\alpha_1, \dots, \alpha_n) = \sum_{i=1}^n E_{\text{stretch}}^i + E_{\text{bend}}^i + E_{\text{endpoints}}^i = \sum_{i=1}^n E_0^i + \alpha_i E^i. \quad (4.3.7)$$

**Structural Stability.** Our goal is to find a set of  $\alpha_i \in \{0,1\}, i = 1, \dots, n$ , within the given budget, such that we maximize the stability of the structure. We define the structural stability of a set of curves via studying its *worst-case load*, a set of external forces, of unit norm in total, applied to the vertices of the structure, that causes the largest displacements  $x$ . Such structural stability is known to be equivalent to computing the smallest eigenvalue of the Hessian  $\mathbf{H} = \mathbf{H}(\alpha_1, \dots, \alpha_n)$  of the elastic energy [Achtziger and Kočvara, 2007], a symmetric semi-positive definite matrix:

$$\mathbf{H}x = \lambda x. \quad (4.3.8)$$

Naively defining the smallest eigenvalue  $\lambda_1(\mathbf{H})$  as the stability of a structure, however, would fail in our setup, since if  $\alpha_i = 0$ , the Hessian  $\mathbf{H}(\alpha_1, \dots, \alpha_n)$  may become singular,

yielding  $\lambda_1(\mathbf{H}) = 0$ . Intuitively, if a curve  $i$  has at least two other curves intersecting it, as soon as  $\alpha_i = 0$ , the curve segment between those intersections becomes disconnected from the rest of the structure and can be moved freely (Fig. 4.6a). Similarly, the segments next to the boundary become free to rotate around a fixed endpoint (Fig. 4.6b). Furthermore, when two intersecting curves  $i, j$  have both  $\alpha_i = \alpha_j = 0$ , their shared vertex becomes disconnected from the overall structure and can be translated in any direction without resistance, again making the Hessian singular.

Instead, following Achtziger and Kočvara [2007], we define the structural stability of our structure for given  $\alpha_1, \dots, \alpha_n$  as the smallest **non-zero** eigenvalue value, which we denote as  $\widetilde{\lambda}_1(\mathbf{H})$ . We first give a formulation of our problem using  $\widetilde{\lambda}_1$ , then give an equivalent formulation using the standard minimum eigenvalue  $\lambda_1$  via Hessian regularization (Sec. 4.3.3).

We first represent the Hessian  $H$  of the total elastic energy as a function of  $\alpha_i$ . Elastic energy is a linear function with respect to  $\alpha_i$ , so the Hessian is also a linear function of the corresponding Hessians:

$$\mathbf{H}(\alpha_1, \dots, \alpha_n) = \sum_{i=1}^n \mathbf{H}_0^i + \sum_{i=1}^n \alpha_i \mathbf{H}^i, \quad (4.3.9)$$

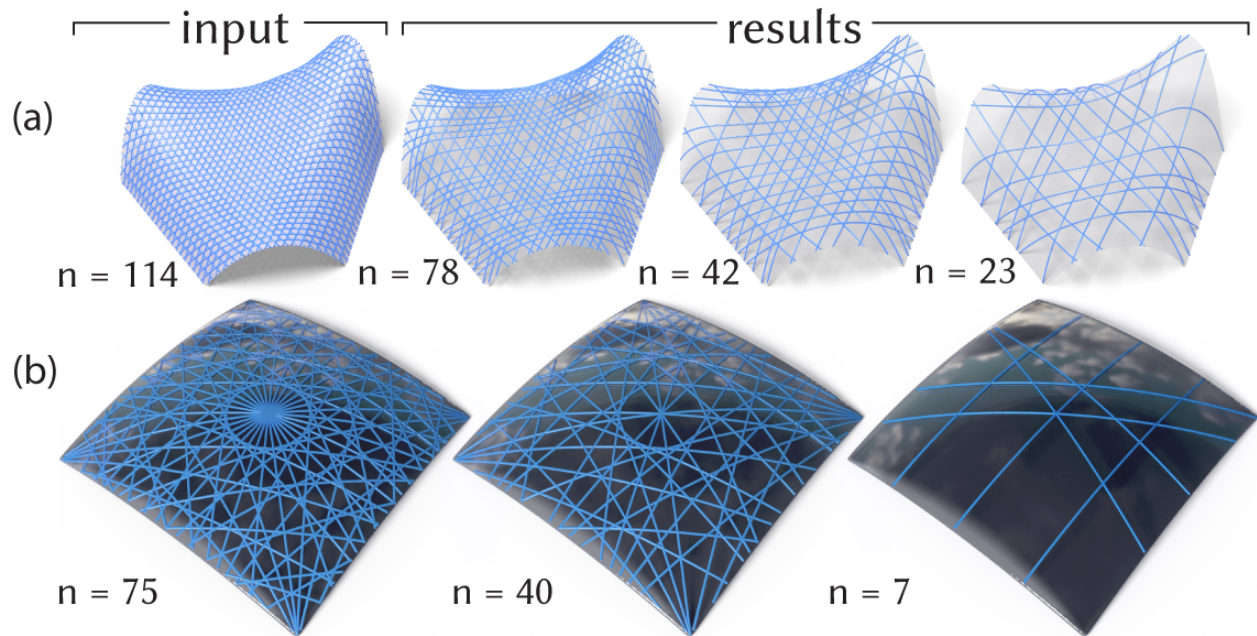
where  $\mathbf{H}^i$  and  $\mathbf{H}_0^i$  are Hessians of  $E^i$  and  $E_0^i$  in Eq. 4.3.7 respectively.

Finally, our curve network simplification can then be formulated as follows:

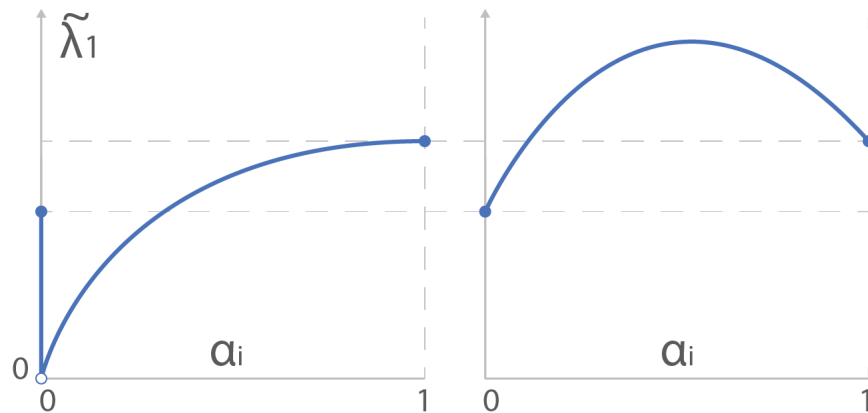
$$\begin{aligned} \max \quad & \widetilde{\lambda}_1 \left( \sum_{i=1}^n \mathbf{H}_0^i + \sum_{i=1}^n \alpha_i \mathbf{H}^i \right) \\ \text{s.t.} \quad & \alpha_1, \dots, \alpha_n \in \{0, 1\} \\ & \sum_{i=1}^n \alpha_i L_i \leq V \end{aligned} \quad (4.3.10)$$

where  $L_i$  is the initial length of the  $i^{\text{th}}$  curve and  $V$  is the user-defined material budget.

The curves with  $\alpha_i = 0$  are considered to be removed from the system, hence we compute the cost of the curve network as  $\sum_{i=1}^n \alpha_i L_i$  for the budget constraint.



**Fig. 4.7.** The user-provided budget constraint can be used to control the final number of curves. Since we store all intermediate results, we allow a user to instantaneously revert to any larger number of curves, if desired.



**Fig. 4.8.** Left: Without Hessian regularization, the relaxation of the binary constraint is discontinuous. Right: Our regularization results in a continuous and concave relaxation with exact  $\lambda$  for binary values of  $\alpha_i$ .

### 4.3.3. Mixed-Integer Semi-definite Programming (MISDP) Formulation

Unfortunately,  $\tilde{\lambda}_1(\mathbf{H}(\alpha_1, \dots, \alpha_n))$  is discontinuous as a function of  $\alpha_i$ , and thus cannot be directly used in a gradient-based optimization (Fig. 4.8 left). Intuitively, as  $\alpha_i$  is approaching 0,  $\tilde{\lambda}_1$  is also decreasing, until it becomes equal to a different eigenvalue when  $\alpha_i = 0$  —



hence the discontinuity. To alleviate this problem, we propose to use Hessian regularization. Namely, we can express the minimum non-zero eigenvalue  $\widetilde{\lambda}_1$  of the original Hessian as the ordinary minimum eigenvalue  $\lambda_1$  of the regularized Hessian (Fig. 4.8 right):

$$\widetilde{\lambda}_1 = \lambda_1 \left( \sum_{i=1}^n \mathbf{H}_0^i + \sum_{i=1}^n [\alpha_i \mathbf{H}^i + (1 - \alpha_i) \mathbf{Q}_i] + \sum_{i,j} \beta_{i,j} \mathbf{Q}_{i,j} \right),$$

where  $\beta_{i,j}$  are auxiliary variables. Here  $\mathbf{Q}_i$  and  $\mathbf{Q}_{i,j}$  are diagonal Tikhonov regularization matrices. We define  $\mathbf{Q}_i$  as 1 on the diagonal for all the vertices of curve  $i$  except for shared intersections, and  $\mathbf{Q}_{i,j}$  as 1 for all the shared intersection vertices between curves  $i$  and  $j$ , and 0 otherwise. Note that since the non-zero column-vectors of our regularizer form a basis of the Hessian null space for  $\alpha_i = 0$ , such regularization preserves the non-zero minimum eigenvalue, i.e., the values of  $\widetilde{\lambda}_1$  for  $\alpha_i \in \{0, 1\}$  are the same with and without regularization, as illustrated by the two dashed horizontal lines in Figure 4.8.

We need to regularize the shared vertices at the intersections of curves  $i, j$  only when both curves are removed, adding  $\beta_{i,j} \mathbf{Q}_{i,j}$ , where  $\beta_{i,j} \in \{0, 1\}$  and  $\beta_{i,j} = 1$  when  $\alpha_i = \alpha_j = 0$ . A naive implementation of this requirement on  $\beta_{i,j}$  would lead to a quadratic constraint and therefore a nonlinear SDP formulation, notoriously hard to solve efficiently. Instead, we observe that for  $\alpha \in \{0, 1\}$  this constraint is equivalent to  $\beta_{i,j} \leq \min(1 - \alpha_i, 1 - \alpha_j)$ , or two linear inequalities.

Finally, finding  $\lambda_1(A)$  is equivalent to a maximization problem  $\max \lambda$  with a semi-definite constraint  $A \succeq \lambda I$  [Vandenberghe and Boyd, 1999]. Using the regularization of the Hessian,

we come to our final formulation of curve network simplification as an instance of mixed-integer semi-definite programming (MI-SDP):

$$\begin{aligned}
& \max_{\alpha} \lambda \\
& \text{s.t. } \sum_{i=1}^n \mathbf{H}_0^i + \sum_{i=1}^n [\alpha_i \mathbf{H}^i + (1 - \alpha_i) \mathbf{Q}_i] + \sum_{i,j} \beta_{i,j} \mathbf{Q}_{i,j} \succeq \lambda I \\
& \alpha_i, \beta_{i,j} \in \{0, 1\} \\
& \beta_{i,j} \leq 1 - \alpha_i; \beta_{i,j} \leq 1 - \alpha_j \\
& \sum_{i=1}^n \alpha_i L_i \leq V.
\end{aligned} \tag{4.3.11}$$

To eliminate the trivial solution  $\alpha_1 = \dots = \alpha_n = 0$ , when there are no curves left and the minimum eigenvalue  $\lambda$  is driven by the regularization, we add a constraint  $\sum_{i=1}^n \alpha_i \geq 1$ .

## 4.4. Solver Mechanism

A natural approach to solving formulation in Eq. 4.3.11 is via its relaxation into an instance of semi-definite programming (SDP) by replacing the integer constraints  $\alpha_i \in \{0, 1\}$  with a continuous constraint  $0 \leq \alpha_i \leq 1$ . This relaxation gives an upper bound to the mixed-integer problem. This can be easily seen from Figure 4.8, right: the function is concave and has maximum inside the interval, not on the endpoints that would correspond to the integer solution  $\alpha_i = 0$  or  $\alpha_i = 1$ . The relaxed solution  $\alpha_{SDP}$  can be either used directly as an approximation via rounding each element to the nearest integer, or as a conservative upper bound for mixed-integer optimization algorithms such as branch-and-bound [Gally et al., 2018].

We tried a standard MI-SDP solver YALMIP [Löfberg, 2004] and implemented a custom branch-and-bound method using MOSEK [ApS, 2020] as the SDP solver. Our experiments showed that SDP relaxation significantly reduces the space of branch-and-bound as compared to an exhaustive search (cutting out 50% to 70% of the search space). Unfortunately, for small to medium numbers of curves, the overhead of solving SDP makes branch-and-bound

significantly slower than the exhaustive search. For instance, for 9 initial curves, branch-and-bound takes several hours, while the exhaustive search is nearly instantaneous. While theoretically for larger numbers exhaustive search should become more expensive, for those numbers of curves both approaches are practically infeasible.

Instead, we propose an efficient and straightforward approximation algorithm. We observe that intuitively, removing any curve from a set can only decrease the stability of the structure. This suggests a process where, starting with the original curve network, we iteratively remove the curve that is the least important for the overall stability, until the budget constraint is satisfied. At each iteration, we compute the equilibrium configuration via the Newton method. We denote the initial set of curves  $S$ , and  $\widetilde{\lambda}_1(A)$  as the minimum non-zero eigenvalue of the Hessian for the structure made of curves in  $A \subseteq S$ . Using Eq. 4.3.9, we set  $\widetilde{\lambda}_1(A) = \widetilde{\lambda}_1(\mathbf{H}(\alpha_1, \dots, \alpha_n))$ , where  $\alpha_i = 1$  for all  $i \in A$ , and 0 otherwise. Finally, denoting  $Vol(A) = \sum_{i \in A} L_i$ , we get Algorithm 1. Here the maximization is performed by simply trying to remove each curve from the current set.

---

**Algorithm 1** Greedy Approximation Algorithm

---

```

Set  $A = S$ ;
while  $Vol(A) > V$  do
    Find  $c' = \arg \max_{c \in A} \widetilde{\lambda}_1(A \setminus \{c\})$ ;
    Set  $A = A \setminus \{c'\}$ ;
end while

```

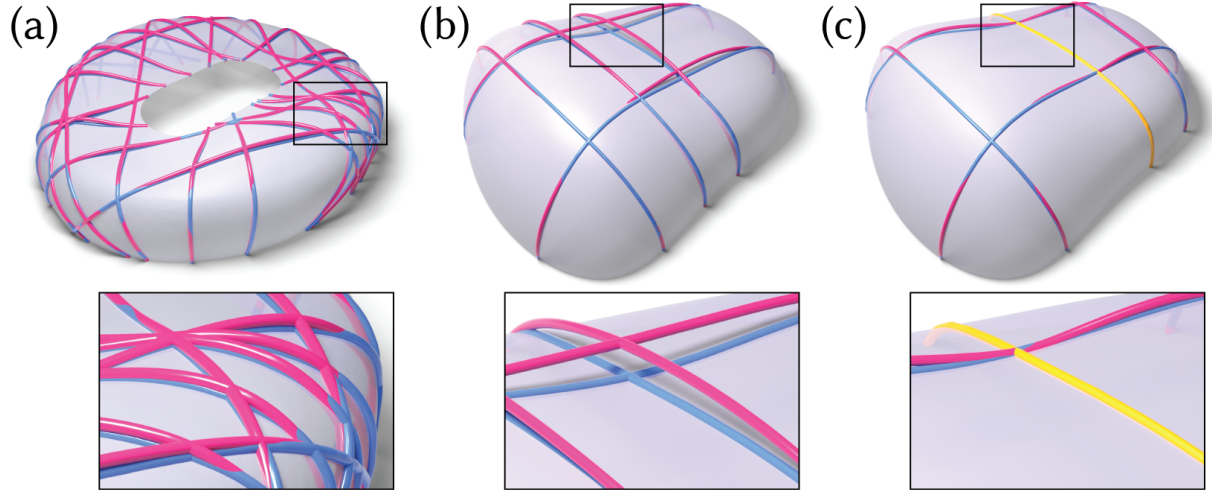
---

Note that while formally  $\widetilde{\lambda}_1(A \setminus \{c\})$  can be larger than  $\widetilde{\lambda}_1(A)$ , e.g., when a rare especially weak curve, for instance long with high curvature, exhibits localized buckling, this does not change or impede the algorithm; no special treatment is needed for such cases.

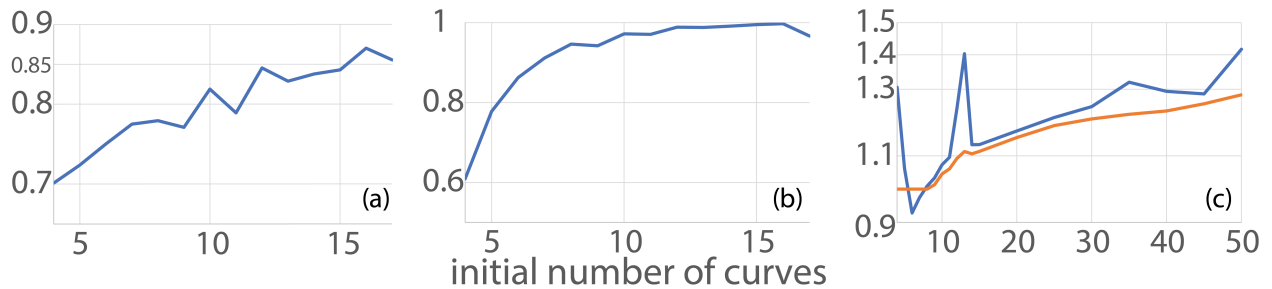
As we discuss in the following section, we find that this simple algorithm approximates the optimal solution with a high ratio, is efficient, is parallelizable per iteration, and outperforms naive approaches.

## 4.5. Results, Validation, and Discussion

Using our method, we have simplified a number of curve networks, depicted in Figures 4.1, 4.7, 4.11, and 4.12. The input surfaces include positive (e.g., tower in Fig. 4.11d) and



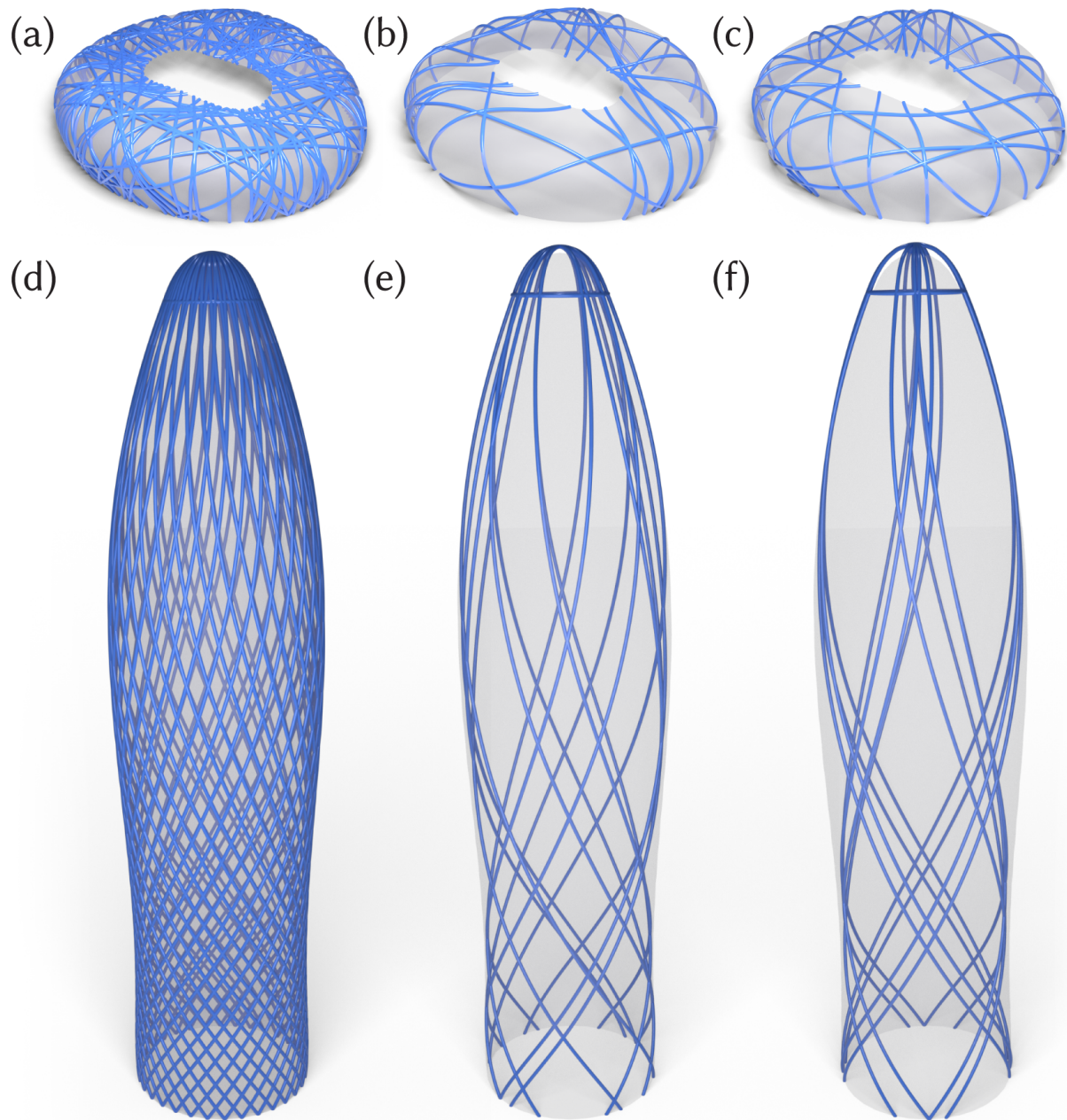
**Fig. 4.9.** Many of our initial networks are nearly at their equilibrium state (e.g., (a), blue: initial curves, pink: the curves at equilibrium). The equilibrium configuration for a simplified curve network might deviate from the target surface (b). For this example, however, adding a single, manually selected, rigid pre-bent curve (c, yellow) is enough to alleviate the deviation.



**Fig. 4.10.** (a) The approximation ratio of our algorithm,  $\lambda_{\text{ours}}/\lambda_{\text{optimum}} > 80\%$  for a sufficient number of curves. (b) Percentile of our solutions among all combinations, typically within the 90 to 99% percentile. (c) The ratio  $\lambda_{\text{ours}}/\lambda_{\text{random search}}$ ; average in blue, median in orange. Our solutions are more stable than random, sometimes by 40%.

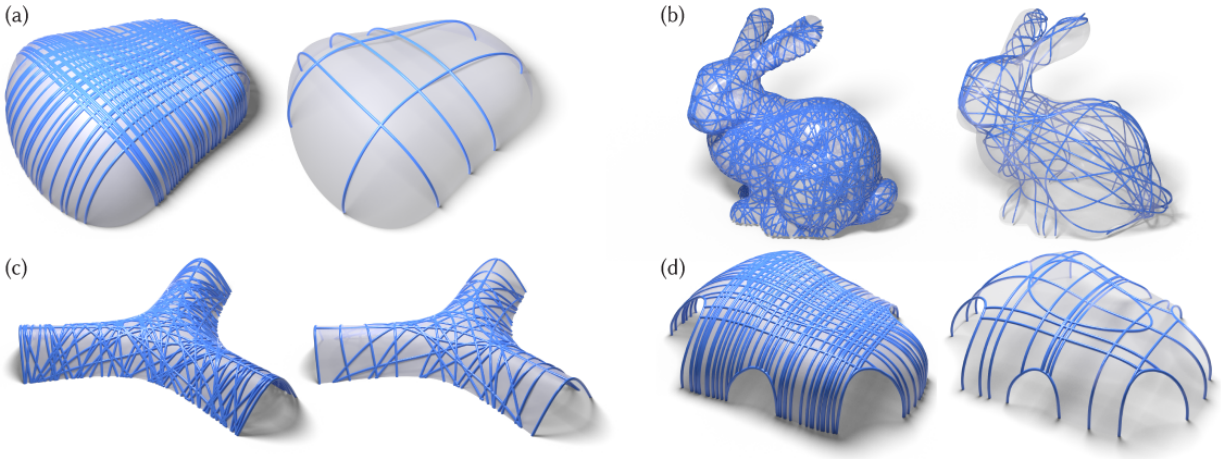
negative Gaussian curvature (e.g., tent, Fig. 4.12b), surfaces with one or two boundaries (stadium in Fig. 4.12d).

**Initial Curve Networks.** The curves on a surface can be modeled via conventional [Autodesk, 2022] or modern interfaces [Arora and Singh, 2020]. To demonstrate our method, we automatically generate random geodesic curves for the bunny (Fig. 4.12b), the hill (Fig. 4.1), the stadium (Fig. 4.11a), and the tent examples (Fig. 4.12c), or trace two- and three-direction frame fields [Panozzo et al., 2014] for the arched and regular shells (Figs. 4.12a,d respectively)



**Fig. 4.11.** Compared to the results of the random search algorithm (b,e), our results (c,f) are significantly more stable (186% for (c), 116% for (f)). For a highly irregular input curve network (a), worst-case stability may not be an intuitive notion (cf. (b, c)). For a regular curve network (d), stability may be related to irregularities and thus is easier to see (cf. (e, f)).

and the kagome pattern example respectively (Fig. 4.7a). We modeled the initial curve network on the tower example (Fig. 4.11d) in a modeling software.



**Fig. 4.12.** A gallery of additional results.

**Table 4.1.** Algorithm statistics for different curve networks.

Surface	Fig.	initial # of curves	initial # of vertices	final # of curves	time (min.)
hill	4.1	73	6319	9	17.7
kagome	4.7a	114	10224	23	53.9
roof	4.7b	75	2813	7	4.6
stadium	4.11a	100	6646	25	16.9
tower	4.11d	33	5699	9	2.5
shell	4.12a	63	5504	5	7.1
bunny	4.12b	75	25442	7	266.0
tent	4.12c	112	6107	42	24.7
arched shell	4.12d	67	3514	21	2.6

To generate a frame field, we use the representation and the optimization of PolyVector fields [Diamanti et al., 2014]. We use PolyVector polynomials of degree 4 or 6, to capture two or three directions per triangle respectively. We then sample the surface boundaries and trace the curves using the classical Euler integration method [Polthier and Schmies, 2006]. To trace the frame fields, we use principal matching, removing any traced curve that hits a frame field singularity. We follow the standard definition of a singularity as either a zero of one of the directions of a field, or an inconsistency of matching directions around a given vertex.

**Equilibrium Configurations.** We explicitly compute the equilibrium configuration for all the curve networks via Newton’s method. Most of the initial configurations are close to the equilibrium (Fig. 4.9 and Supplementary). The final curve networks at their equilibrium

might deviate from the target surface (e.g., Fig. 4.9b). If undesired, this effect can be reduced for fabrication by adding pre-bent stiff curves to the initial curve network (yellow curve in Fig. 4.9c).

**Quantitative Evaluation.** We numerically compare our algorithm against two baseline algorithms: exhaustive and random searches.

The exhaustive search simply tries all subsets of the given curves and selects the most stable combination within a budget. Clearly, the algorithm has exponential complexity, yet produces the true optimum. We plotted the approximation ratio of our algorithm compared to this true optimum, i.e., the ratio of  $\lambda_{\text{ours}}$ , the stability of our result, and  $\lambda_{\text{optimum}}$  in Fig. 4.10a. We tested 200 random curves on each of four simple test surfaces and averaged the ratios. We were able to run the exhaustive search in a reasonable time for 17 curves maximum. As seen from the plot, our approximation ratio is increasing with the number of curves and reaches roughly 85%. For comparison, for 19 curves, our algorithm takes less than 1 second, while exhaustive search takes around 45 minutes.

We furthermore display the percentile of our solution among all the combinations (Fig. 4.10b). Our solution is always in the top 90 – 99% percentile for a sufficient number of curves.

Finally, we compare our algorithm to a random search algorithm, which generates uniformly distributed random combinations within the budget and chooses the most stable one using the definition in Eq. 4.3.10. For a fair comparison, we run the random search for the same time as our algorithm. The ratio of  $\lambda_{\text{ours}}$  divided by the best  $\lambda$  of the random search is presented in Fig. 4.10c. We run this experiment 400 times, each time generating a fixed number of random geodesics on a given surface, and display average (blue) and median values (orange). As indicated by the plot, our algorithmic solution is roughly 20% to 40% more stable than the solution obtained by the random search, and this ratio increases as the space of possible combinations grows.

**Performance.** The performance of our algorithm depends on the number of the initial curves, the material budget, and the vertex sampling density of the polylines. On a desktop

computer, our computation time typically varies from 2 minutes for the tower (Fig. 4.11f) to 1 hour for the kagome pattern example (Fig. 4.7a). The only exception is the bunny (Fig. 4.12b) that took 4.5 hours. See Table 4.1 for the full statistics.

**Limitations and Future Work.** A natural extension of our work is to optimize the shapes of the selected curves to further improve the network’s stability while preserving the artist intent; we only focus on selecting a subset of curves with fixed geometry. Another exciting direction is to perform a simultaneous simplification and stylization of curve networks, akin to sketch stylization.



# Chapter 5

---

## Conclusion

In this master's thesis, we explored the rationalization of free-form surfaces through curve networks. Our goal was to give designers more freedom, as automatic tools were mostly limited to gridshells. Therefore, our aspiration was to facilitate the design of irregular curve networks. It's in this optic that we presented a novel formulation and an efficient method for simplification of 3D curve networks focusing on their worst-case stability. Our method is compatible with standard interfaces for creating 3D curves and thus has immediate applications in fabrication-aware modeling. We hope that it will form the core of a future fabrication-aware design system allowing users to interactively create, edit, and simplify curve networks. An implementation of this algorithm is available as an open-source software (<https://github.com/wwwnev/Stability-Aware-Simplification-of-Curve-Networks>).

Following the work presented in this master's thesis, many research directions remain open. First, the project could be directly expanded. For example, it could easily be adapted to include the general formulation for discrete elastic rods model from Bergou et al. [2008, 2010], allowing for anisotropic rods. Although this model does not hold that far, rods could also be replaced by strips of materials such as paper or metal. Furthermore, the energy used in the greedy algorithm (Alg. 1) could be enhanced, for example taking into account the distance between the curve network at equilibrium and the free-form surface [Panozzo et al., 2013; Vouga et al., 2012].

Another avenue would be to salvage the MI-SDP optimization model presented in Section 4.3. This would be an important improvement over our work. While we have been unsuccessful in approximating satisfactory integer solutions by relaxing the integer constraint, future work in MI optimization approximation might prove useful to this end.

While we limited ourselves to topology optimization, geometric optimization could be a very promising avenue of research. By allowing its curves to move or slither across the free-form surface, the stability of a curve network could undoubtedly be further improved. The original idea for this master's thesis, and associated article, was to solve the problem of stable curve networks using the two approaches, topological and geometrical optimization, in a single heuristic. In the end, aspirations had to be scaled back and the focus was put on topology only. However, part of the presented framework, namely the curve network representation and energy formulation, could also be used in geometrical optimization.

## References

---

- Wolfgang Achtziger and Michal Kočvara. 2007. Structural topology optimization with eigenvalues. *SIAM Journal on Optimization* 18, 4 (2007), 1129–1164. <https://doi.org/10.1137/060651446>
- Ergun Akleman, Jianer Chen, Qing Xing, and Jonathan L. Gross. 2009. Cyclic Plain-Weaving on Polygonal Mesh Surfaces with Graph Rotation Systems. *ACM Trans. Graph.* 28, 3, Article 78 (jul 2009), 8 pages. <https://doi.org/10.1145/1531326.1531384>
- MOSEK ApS. 2020. *The MOSEK Fusion API for C++ manual. Version 8.1.* <https://docs.mosek.com/8.1/cxxfusion/index.html>
- Rahul Arora, Alec Jacobson, Timothy R. Langlois, Yijiang Huang, Caitlin Mueller, Wojciech Matusik, Ariel Shamir, Karan Singh, and David I.W. Levin. 2019. Volumetric michell trusses for parametric design & fabrication. *Proceedings: SCF 2019 - ACM Symposium on Computational Fabrication* (2019). <https://doi.org/10.1145/3328939.3328999>
- Rahul Arora and Karan Singh. 2020. Mid-Air Drawing of Curves on 3D Surfaces in AR/VR. *CoRR* abs/2009.09029 (2020). arXiv:2009.09029 <https://arxiv.org/abs/2009.09029>
- Autodesk. 2022. Maya 2022. <https://autodesk.com/maya>
- A. Ben-Tal and A. Nemirovski. 1997. Robust truss topology design via semidefinite programming. *SIAM Journal on Optimization* 7, 4 (1997), 991–1016. <https://doi.org/10.1137/S1052623495291951>
- Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2010. Discrete Viscous Threads. *ACM Trans. Graph.* 29, 4, Article 116 (jul 2010), 10 pages. <https://doi.org/10.1145/1778765.1778853>

- Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete Elastic Rods. *ACM Transactions on Graphics (SIGGRAPH)* 27, 3 (aug 2008), 63:1–63:12.
- Gaurav Bharaj, David I. W. Levin, James Tompkin, Yun Fei, Hanspeter Pfister, Wojciech Matusik, and Changxi Zheng. 2015. Computational Design of Metallophone Contact Sounds. *ACM Trans. Graph.* 34, 6, Article 223 (oct 2015), 13 pages. <https://doi.org/10.1145/2816795.2818108>
- Pengbo Bo, Helmut Pottmann, Martin Kilian, Wenping Wang, and Johannes Wallner. 2011. Circular arc structures. *ACM Transactions on Graphics* 30, 4 (2011), 1–12. <https://doi.org/10.1145/1964921.1964996>
- Marcel Campen and Leif Kobbelt. 2014. Dual Strip Weaving: Interactive Design of Quad Layouts Using Elastica Strips. *ACM Trans. Graph.* 33, 6, Article 183 (nov 2014), 10 pages. <https://doi.org/10.1145/2661229.2661236>
- Adelaide Cerveira, Agostinho Agra, Fernando Bastos, and Joaquim A. S. Gromicho. 2010. New branch and bound approaches for truss topology design with discrete areas.
- Desai Chen, David I. W. Levin, Wojciech Matusik, and Danny M. Kaufman. 2017. Dynamics-Aware Numerical Coarsening for Fabrication Design. *ACM Trans. Graph.* 36, 4, Article 84 (jul 2017), 15 pages. <https://doi.org/10.1145/3072959.3073669>
- Elena Cherkaev and Andrej Cherkaev. 2004. *Principal Compliance and Robust Optimal Design*. Springer Netherlands, Dordrecht, 169–196. [https://doi.org/10.1007/1-4020-2308-1\\_14](https://doi.org/10.1007/1-4020-2308-1_14)
- Qiaodong Cui, Timothy Langlois, Pradeep Sen, and Theodore Kim. 2020. Fast and Robust Stochastic Structural Optimization. *Computer Graphics Forum* 39, 2 (2020), 385–397. <https://doi.org/10.1111/cgf.13938>  
arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13938>
- Bernardino D’Amico, A. Kermani, Hexin Zhang, Alberto Pugnale, Sofia Colabella, and Sergio Pone. 2015. Timber gridshells: Numerical simulation, design and construction of a full scale structure. *Structures* 3 (06 2015). <https://doi.org/10.1016/j.istruc.2015.>

- Fernando de Goes, Pierre Alliez, Houman Owhadi, and Mathieu Desbrun. 2013. On the Equilibrium of Simplicial Masonry Structures. *ACM Trans. Graph.* 32, 4, Article 93 (jul 2013), 10 pages. <https://doi.org/10.1145/2461912.2461932>
- Frederic de Gournay, Grégoire Allaire, and François Jouve. 2008. Shape and topology optimization of the robust compliance via the level set method. *ESAIM: COCV* 14, 1 (2008), 43–70. <https://doi.org/10.1051/cocv:2007048>
- Olga Diamanti, Daniele Panozzo, and Olga Sorkine-hornung. 2014. Designing N-PolyVector Fields with Complex Polynomials. *Eurographics Symposium on Geometry Processing* 33, 5 (2014).
- William S. Dorn, Ralph E. Gomory, and Harvey J. Greenberg. 1964. Automatic design of optimal structures.
- Michael Eigensatz, Martin Kilian, Alexander Schiftner, Niloy J. Mitra, Helmut Pottmann, and Mark Pauly. 2010. Paneling architectural freeform surfaces. *ACM SIGGRAPH 2010* 1, 212 (2010), 1–10. <https://doi.org/10.1145/1778765.1778782>
- Koos Eissen and Roselien Steur. 2007. *Sketching : drawing techniques for product designers* (paperback ed.). BIS Publishers, Amsterdam, Netherlands.
- Chi Wing Fu, Chi Fu Lai, Ying He, and Daniel Cohen-Or. 2010. K-set tilable surfaces. *ACM Transactions on Graphics* 29, 4 (2010), 1–6. <https://doi.org/10.1145/1778765.1778781>
- Tristan Gally, Marc E. Pfetsch, and Stefan Ulbrich. 2018. A framework for solving mixed-integer semidefinite programs. *Optimization Methods and Software* 33, 3 (2018), 594–632. <https://doi.org/10.1080/10556788.2017.1322081>  
arXiv:<https://doi.org/10.1080/10556788.2017.1322081>
- Akash Garg, Andrew O. Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. 2014. Wire Mesh Design. *ACM Trans. Graph.* 33, 4 (2014), 66:1–66:12. <https://doi.org/10.1145/2601097.2601106>

- Caigui Jiang, Chengcheng Tang, Hans Peter Seidel, Renjie Chen, and Peter Wonka. 2019. Computational design of lightweight trusses. *arXiv* (2019). arXiv:1901.05637
- Mina Konaković-Luković, Julian Panetta, Keenan Crane, and Mark Pauly. 2018. Rapid deployment of curved surfaces via programmable auxetics. *ACM Transactions on Graphics* 37, 4 (2018), 1–13. <https://doi.org/10.1145/3197517.3201373>
- Shahar Z. Kovalsky, Noam Aigerman, Ronen Basri, and Yaron Lipman. 2014. Controlling Singular Values with Semidefinite Programming. *ACM Trans. Graph.* 33, 4, Article 68 (jul 2014), 13 pages. <https://doi.org/10.1145/2601097.2601142>
- Michal Kočvara. 2015. Truss topology design with integer variables made easy. January 2010 (2015). [http://www.optimization-online.org/DB\\_FILE/2010/05/2614.pdf](http://www.optimization-online.org/DB_FILE/2010/05/2614.pdf)
- Timothy Langlois, Ariel Shamir, Daniel Dror, Wojciech Matusik, and David I. W. Levin. 2016. Stochastic Structural Analysis for Context-Aware Design and Fabrication. *ACM Trans. Graph.* 35, 6, Article 226 (nov 2016), 13 pages. <https://doi.org/10.1145/2980179.2982436>
- Julian Lienhard and Jan Knippers. 2015. Bending-active structures. *Bautechnik* 92, 6 (2015), 394–402. <https://doi.org/10.1002/bate.201500007>
- Hsueh-Ti Derek Liu, Alec Jacobson, and Maks Ovsjanikov. 2019. Spectral Coarsening of Geometric Operators. *ACM Trans. Graph.* 38, 4, Article 105 (jul 2019), 13 pages. <https://doi.org/10.1145/3306346.3322953>
- Yang Liu, Hao Pan, John Snyder, Wenping Wang, and Baining Guo. 2013. Computing self-supporting surfaces by regular triangulation. *ACM Transactions on Graphics* 32, 4 (2013). <https://doi.org/10.1145/2461912.2461927>
- J. Löfberg. 2004. YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *In Proceedings of the CACSD Conference*. Taipei, Taiwan.
- Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. 2014. Build-to-Last: Strength to Weight 3D Printed Objects. *ACM Trans. Graph.* 33, 4, Article 97 (jul 2014), 10 pages. <https://doi.org/10.1145/2601097.2601168>

- Robert McNeel et al. 2010. Rhinoceros 3D, Version 6.0. *Robert McNeel & Associates, Seattle, WA* (2010).
- Kristo Mela. 2014. Resolving Issues with Member Buckling in Truss Topology Optimization Using a Mixed Variable Approach. *Struct. Multidiscip. Optim.* 50, 6 (dec 2014), 1037–1049. <https://doi.org/10.1007/s00158-014-1095-x>
- Eder Miguel, Mathias Lepoutre, and Bernd Bickel. 2016. Computational Design of Stable Planar-Rod Structures. *ACM Transactions on Graphics (SIGGRAPH 2016)* 35, 4 (2016).
- Przemyslaw Musialski, Christian Hafner, Florian Rist, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2016. Non-Linear Shape Optimization Using Local Subspace Projections. *ACM Transactions on Graphics* 35, 4 (2016), 87:1–87:13. <https://doi.org/10.1145/2897824.2925886>
- Julian Panetta, Mina Konaković-Luković, Florin Isvoranu, Etienne Bouleau, and Mark Pauly. 2019. X-Shells: A new class of deployable beam structures. *ACM Transactions on Graphics* 38, 4 (2019). <https://doi.org/10.1145/3306346.3323040>
- Julian Panetta, Abtin Rahimian, and Denis Zorin. 2017. Worst-Case Stress Relief for Microstructures. *ACM Trans. Graph.* 36, 4, Article 122 (jul 2017), 16 pages. <https://doi.org/10.1145/3072959.3073649>
- Daniele Panozzo, Philippe Block, and Olga Sorkine-Hornung. 2013. Designing unreinforced masonry models. *ACM Transactions on Graphics* 32, 4 (2013). <https://doi.org/10.1145/2461912.2461958>
- D Panozzo, E Puppo, M Tarini, and O Sorkine-Hornung. 2014. Frame Fields: Anisotropic and Non-Orthogonal Cross Fields. *Acm Transactions on Graphics* 33, 4 (2014), 11. <https://doi.org/10.1145/2601097.2601179>
- Davide Pellis, Martin Kilian, Helmut Pottmann, and Mark Pauly. 2021. Computational design of weingarten surfaces. *ACM Transactions on Graphics* 40, 4 (2021). <https://doi.org/10.1145/3450626.3459939>
- Jesús Pérez, Miguel A. Otaduy, and Bernhard Thomaszewski. 2017. Computational Design and Automated Fabrication of Kirchhoff-Plateau Surfaces. *ACM Trans. Graph.* 36, 4,

- Article 62 (jul 2017), 12 pages. <https://doi.org/10.1145/3072959.3073695>
- Stefan Pillwein and Przemyslaw Musialski. 2021. Generalized Deployable Elastic Geodesic Grids. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2021)* 40, 6 (2021), in press. <https://doi.org/10.1145/3478513.3480516> arXiv:arXiv:2111.08883v1
- Konrad Polthier and Markus Schmies. 2006. Straightest Geodesics on Polyhedral Surfaces. In *ACM SIGGRAPH 2006 Courses (Boston, Massachusetts) (SIGGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 30–38. <https://doi.org/10.1145/1185657.1185664>
- Ivan Puhachov, William Neveu, Edward Chien, and Mikhail Bessmeltsev. 2021. Keypoint-Driven Line Drawing Vectorization via PolyVector Flow. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 40, 6 (Dec. 2021). <https://doi.org/10.1145/3478513.3480529>
- Jesus Pérez, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, José A. Canabal, Robert Sumner, and Miguel A. Otaduy. 2015. Design and Fabrication of Flexible Rod Meshes. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)* 34, 4 (2015). <http://www.gmrv.es/Publications/2015/PTCBCS015>
- Yingying Ren, Switzerland Julian Panetta, Uc Davis, Usa Tian Chen, Switzerland Florin Isvoranu, Switzerland Samuel Poincloux, Switzerland Christopher Brandt, Switzerland Alison Martin, Independent Researcher, Italy Mark Pauly, Julian Panetta, Tian Chen, Florin Isvoranu, Samuel Poincloux, Christopher Brandt, Alison Martin, and Mark Pauly. 2021. 3D Weaving with Curved Ribbons. *ACM Transactions on Graphics* 40, 4 (2021), 1–15. <https://doi.org/10.1145/3450626.3459788>
- Eike Schling, Wang Hui, Jonas Schikore, and Helmut Pottmann. 2018. Design and Construction of Curved Support Structures with Repetitive Parameters.
- Christian Schumacher, Jonas Zehnder, and Moritz Bächer. 2018. Set-in-Stone: Worst-Case Optimization of Structures Weak in Tension. *ACM Trans. Graph.* 37, 6, Article 252 (dec 2018), 13 pages. <https://doi.org/10.1145/3272127.3275085>



- Christian Schüller, Roi Poranne, and Olga Sorkine-Hornung. 2018. Shape representation by zippables. *ACM Transactions on Graphics* 37, 4 (2018). <https://doi.org/10.1145/3197517.3201347>
- Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. 2012. Stress Relief: Improving Structural Strength of 3D Printable Objects. *ACM Trans. Graph.* 31, 4, Article 48 (jul 2012), 11 pages. <https://doi.org/10.1145/2185520.2185544>
- Mathias Stolpe. 2016. Truss optimization with discrete design variables: a critical review. *Structural and Multidisciplinary Optimization* 53, 2 (2016), 349–374. <https://doi.org/10.1007/s00158-015-1333-x>
- Masahito Takezawa, Takuma Imai, Kentaro Shida, and Takashi Maekawa. 2016. Fabrication of Freeform Objects by Principal Strips. *ACM Trans. Graph.* 35, 6, Article 225 (nov 2016), 12 pages. <https://doi.org/10.1145/2980179.2982406>
- Ye Tao, Nannan Lu, Caowei Zhang, Guanyun Wang, Cheng Yao, and Fangtian Ying. 2016. CompuWoven: A Computer-Aided Fabrication Approach to Hand-Woven Craft. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (San Jose, California, USA) (*CHI EA '16*). Association for Computing Machinery, New York, NY, USA, 2328–2333. <https://doi.org/10.1145/2851581.2892293>
- Ye Tao, Guanyun Wang, Caowei Zhang, Nannan Lu, Xiaolian Zhang, Cheng Yao, and Fangtian Ying. 2017. *WeaveMesh: A Low-Fidelity and Low-Cost Prototyping Approach for 3D Models Created by Flexible Assembly*. Association for Computing Machinery, New York, NY, USA, 509–518. <https://doi.org/10.1145/3025453.3025699>
- Toshikazu Torigaki, Ichiro Hagiwara, Yuichi Kitagawa, Maki Ueda, Zheng-Dong Ma, and Noboru Kikuchi. 1994. Development and Application of a Shape-Topology Optimization System Using a Homogenization Method. *SAE Transactions* 103 (1994), 1217–1223. <http://www.jstor.org/stable/44611836>
- Lieven Vandenbergh and Stephen Boyd. 1999. Applications of semidefinite programming. *Applied Numerical Mathematics* 29, 3 (1999), 283–299.

- Amir Vaxman, Christian Müller, and Ofir Weber. 2017. Regular meshes from polygonal patterns. *ACM Transactions on Graphics* 36, 4 (2017). <https://doi.org/10.1145/3072959.3073593>
- Josh Vekhter, Jiacheng Zhuo, Luisa F.Gil Fandino, Qixing Huang, and Etienne Vouga. 2019. Weaving geodesic foliations. *ACM Transactions on Graphics* 38, 4 (2019). <https://doi.org/10.1145/3306346.3323043>
- Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. 2012. Design of Self-Supporting Surfaces. *ACM Trans. Graph.* 31, 4, Article 87 (jul 2012), 11 pages. <https://doi.org/10.1145/2185520.2185583>
- Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. 2016. Designing structurally-sound ornamental curve networks. *ACM Transactions on Graphics* 35, 4 (2016). <https://doi.org/10.1145/2897824.2925888>
- Qingnan Zhou, Julian Panetta, and Denis Zorin. 2013. Worst-Case Structural Analysis. *ACM Trans. Graph.* 32, 4, Article 137 (jul 2013), 12 pages. <https://doi.org/10.1145/2461912.2461967>