

# **Geometric Rationalization for Freeform Architecture**

Ph.D Dissertation by  
**Caigui Jiang**

In Partial Fulfillment of the Requirements

For the Degree of

**Doctor of Philosophy**

King Abdullah University of Science and Technology, Thuwal,  
Kingdom of Saudi Arabia

June, 2016

The thesis of Caigui Jiang is approved by the examination committee

Committee Chairperson: Helmut Pottmann

Committee Co-Chairperson: Peter Wonka

Committee Member: Bernard Ghanem

Committee Member: Mark Pauly

Copyright ©2016

Caigui Jiang

All Rights Reserved

# ABSTRACT

## Geometric Rationalization for Freeform Architecture

Caigui Jiang

The emergence of freeform architecture provides interesting geometric challenges with regards to the design and manufacturing of large-scale structures. To design these architectural structures, we have to consider two types of constraints. First, aesthetic constraints are important because the buildings have to be visually impressive. Second, functional constraints are important for the performance of a building and its efficient construction. This thesis contributes to the area of *architectural geometry*. Specifically, we are interested in the *geometric rationalization* of freeform architecture with the goal of combining aesthetic and functional constraints and construction requirements. Aesthetic requirements typically come from designers and architects. To obtain visually pleasing structures, they favor smoothness of the building shape, but also smoothness of the visible patterns on the surface. Functional requirements typically come from the engineers involved in the construction process. For example, covering freeform structures using planar panels is much cheaper than using non-planar ones. Further, constructed buildings have to be stable and should not collapse. In this thesis, we explore the geometric rationalization of freeform architecture using four specific example problems inspired by real life applications. We achieve our results by developing optimization algorithms and a theoretical study of the underlying geometrical structure of the problems. The four example prob-

lems are the following: (1) The design of shading and lighting systems which are torsion-free structures with planar beams based on quad meshes. They satisfy the functionality requirements of preventing light from going inside a building as shading systems or reflecting light into a building as lighting systems. (2) The Design of freeform honeycomb structures that are constructed based on hex-dominant meshes with a planar beam mounted along each edge. The beams intersect without torsion at each node and create identical angles between any two neighbors. (3) The design of polyhedral patterns on freeform surfaces, which are aesthetic designs created by planar panels. (4) The design of space frame structures that are statically-sound and material-efficient structures constructed by connected beams. Rationalization of cross sections of beams aims at minimizing production cost and ensuring force equilibrium as a functional constraint.

# ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Helmut Pottmann for his help and support during my Ph.D. study. He guided me into the world of geometry and shown me the beauty of research. I would like to express my appreciation to Peter Wonka as my co-advisor, not only for his guidance and encourage on my research, but also plenty of help and suggestions on other aspects. I am grateful to Bernard Ghanem and Mark Pauly for serving as my dissertation committee members, thank you for your valuable suggestions. I would also like to thank Johannes Wallner for his advice on some of my projects. I really appreciate working with him. I truly thank Liu Yang for offering me a great opportunity as a summer intern at Microsoft Research Asia.

My research work would not have been completed without the help from my cooperators. I would express my thanks to Jun Wang, Liangliang Nan, Chengcheng Tang, Xiang Sun, Amir Vaxman, Chi-Han Peng, Philippe Bompas, Michael Barton and Marko Tomicic.

Thanks to Visual Computing Center for its comfortable working environment and its excellent members around. Especially, I would like to thank Liliana Rivera, Tina Smith, Rahman Hasan, Charlotte Rakhit, Tagreed Khalil for their help on traveling and IT assistance. Also thank my friends and colleagues in this big family, including, but not limited to, Dongming Yan, Yongliang Yang, Han Liu, Lubin Fan, Qiang Fu, Ling Shi, Youyi Zheng, Pengbo Bo, Yanchao Yang, Xin Zhao, Baoyuan Wu, Guangming Zang, Jing Ren, Feilong Yan, Tianzhu Zhang, Niloy Mitra, Wolfgang Heidrich, Markus Hadwiger, Ganesh Sundaramoorthi, Jens Schneider, Neil Smith,

Mahmed Ibrahim, Sawsan AlHalawani.

I gratefully acknowledge the financial support from KAUST Fellowship and related research funding.

Above all, I wish to express my love and heartfelt thanks to my parents, my wife Wen Yang for their continuous love, patient, and support, my daughter Ziling Jiang for her lovely smile.

# TABLE OF CONTENTS

<b>Examination Committee Approval</b>	<b>2</b>
<b>Copyright</b>	<b>3</b>
<b>Abstract</b>	<b>4</b>
<b>Acknowledgements</b>	<b>6</b>
<b>1 Introduction</b>	<b>11</b>
<b>2 Shading and lighting systems</b>	<b>16</b>
2.1 Introduction . . . . .	16
2.2 Line Congruences . . . . .	18
2.2.1 Smooth line congruences . . . . .	18
2.2.2 Congruences defined over triangle meshes . . . . .	21
2.2.3 Congruences defined over quad meshes . . . . .	22
2.3 Normal congruences . . . . .	24
2.3.1 Smooth normal congruences . . . . .	24
2.3.2 Discrete normal congruences . . . . .	25
2.4 Applications and Algorithms . . . . .	27
2.4.1 Optimization of Mesh-based Congruences . . . . .	29
2.4.2 Conversion to Quad-based Torsal Form . . . . .	32
2.4.3 Results . . . . .	34
2.5 Discussion . . . . .	37
2.6 Conclusion . . . . .	38
<b>3 Freeform Honeycomb Structures</b>	<b>42</b>
3.1 Introduction . . . . .	42
3.2 Geometry and flexibility of honeycomb structures . . . . .	45
3.3 Computational approach . . . . .	51
3.4 Results . . . . .	54



3.5	Discussion . . . . .	61
3.6	Conclusion . . . . .	62
<b>4</b>	<b>Polyhedral Patterns</b>	<b>65</b>
4.1	Introduction . . . . .	65
4.2	Previous Work . . . . .	67
4.3	Geometry of Polyhedral Patterns . . . . .	69
4.3.1	Discretization of osculating paraboloids . . . . .	70
4.3.2	Fitting tiles to paraboloids . . . . .	73
4.3.3	Strip decompositions . . . . .	76
4.3.4	Regularizers Motivated by Symmetries . . . . .	78
4.4	Overview and User Interaction . . . . .	79
4.5	Optimization Framework . . . . .	80
4.6	Regularization with Affine Symmetries . . . . .	82
4.6.1	Affine symmetries . . . . .	83
4.6.2	Symmetry in a tangential projection . . . . .	85
4.6.3	Avoiding self-intersection . . . . .	86
4.6.4	Edge length regularization . . . . .	86
4.7	Results . . . . .	87
4.8	Conclusions . . . . .	92
<b>5</b>	<b>Space Frame Structures</b>	<b>95</b>
5.1	Introduction . . . . .	95
5.2	Previous Work . . . . .	98
5.3	Overview . . . . .	100
5.4	Optimization Framework . . . . .	102
5.4.1	Connectivity Enumeration, Ranking, and Editing . . . . .	103
5.4.2	Force Initialization . . . . .	104
5.4.3	Optimization of Node Positions . . . . .	105
5.4.4	Discrete Optimization of Beam Cross Sections . . . . .	108
5.5	Results and Discussion . . . . .	114
5.6	Conclusion . . . . .	119
<b>6</b>	<b>Conclusion</b>	<b>120</b>
6.1	Summary . . . . .	120
6.2	Future Work . . . . .	121

<b>References</b>	<b>123</b>
<b>Appendices</b>	<b>130</b>

# Chapter 1

## Introduction

An increased use of freeform surfaces has changed the map of contemporary architecture in the past decades. On the one hand, this striking trend is boosted by the increasing demand for freeform design. Freeform architecture often creates landmark buildings with unique and stunning visual features, rather than standard and conventional box-like appearances. On the other hand, the cost of achieving freeform architecture is constantly reduced by the development of new technologies. One set of technologies simplify the designing and modeling processing. For example, 3D digital modeling tools can help people obtain almost any imaginable freeform geometry and visualize the building with lighting, temperature and wind simulation before construction. Modern fabrication techniques, especially with the assistance of robotics, make customized manufacturing accessible and affordable. The development of new materials provide more choices in the planning and construction of freeform buildings.

Compared with conventional buildings, the construction cost of freeform buildings is still much higher. One important reason for this high cost is the varying shape of different elements including beams, nodes, panels and so on. In the past decade, the arising new research area of *architectural geometry* [1] sought to obtain a balance between purely freeform design and fabrication efficiency. Such categories of work are called *rationalization* in architecture. In other words, the goal is to discretize a freeform architecture with elements that are suitable for manufacturing, for ex-

ample, planar beams and panels, congruent nodes and bars, and torsion-free nodes. Pottmann et al. summarized recent work on this topic in a survey [2].

In the community of computer graphics and geometry processing, the research work on geometric rationalization is a combination of discrete differential geometry, geometry processing algorithms, and optimization techniques. In this thesis, we explore the geometric rationalization of freeform architecture using four specific example problems. We achieve our results by developing optimization algorithms and a theoretical study of the underlying geometrical structure of the problems. In the following we briefly explain the four rationalization problems that we tackle in this thesis. These problems are motivated by real-world architectural design challenges.

**Shading systems and lighting systems.** They are usually external decorative structures and therefore their functionality is closely related to sunlight. Shading systems are used to prevent light from going inside a building while lighting systems aim to increase the amount of sunlight entering a building by reflection or refraction. In our research, we focus on shading and lighting systems that consist of planar beams arranged along the edges of a quad-dominant base mesh. The mesh itself covers a reference freeform surface. One construction requirement that we use for these structures is that the beams have to be planar and the nodes have to be torsion-free. To achieve this goal, we draw from the theory of line congruences. We derive constraints of discrete line congruences to construct a good initialization of the quad mesh and the node directions at each vertex. Figure 1.1 shows the light simulation for two shading systems constructed by torsion-free structures.

**Honeycomb structures.** Motivated by the structures of beehives, a freeform honeycomb structure is defined as a special torsion-free structure based on a hexdominant mesh on freeform surfaces. Besides the geometrical constraint of the structure being torsion-free, each angle between neighboring planar beams is required to be 120 degrees. Then the nodes at all vertices are congruent with each other and can be

manufactured by cutting a single type of node structure. An honeycomb structure approximated the Cour Visconti roof in the Louvre Museum is shown in figure 1.2.

**Polyhedral patterns.** Polyhedral patterns are polygonal meshes whose faces are all planar. The planarity requirement is one of the primary rationalization goals in architecture. We focus on regular and semi-regular patterns that can be derived from triangle, quad, or hex meshes by geometric subdivision rules. Covering a freeform surface by polyhedral patterns is not trivial, especially at the negative curvature regions. Affine symmetries with respect to axes or planes are proposed as regularizers in the geometric optimization with the constraints of planarity. Polyhedral patterns on a knot model are shown in figure 1.3.

**Space frame structures.** In architecture and structural engineering, a space frame structure is a truss-like, lightweight rigid structure constructed from interlocking struts in a geometric pattern. We propose a systematic computational framework for the design of space structures incorporating considerations like static soundness, approximation of reference surfaces, boundary alignment, and geometric regularity. In addition, we consider the reduction of construction cost by minimizing the total volume of material used for beams under the practical consideration that the cross section can be customized but only a limited number of cross sections are allowed. Formulating these requirements leads to a mixed-integer programming problem with a bilinear objective function and quadratic constraints. To tackle this challenging problem, we propose a novel and practical alternating direction method based on linear programming relaxation. A space structure constructed with six types of customized beams is shown in figure 1.4.

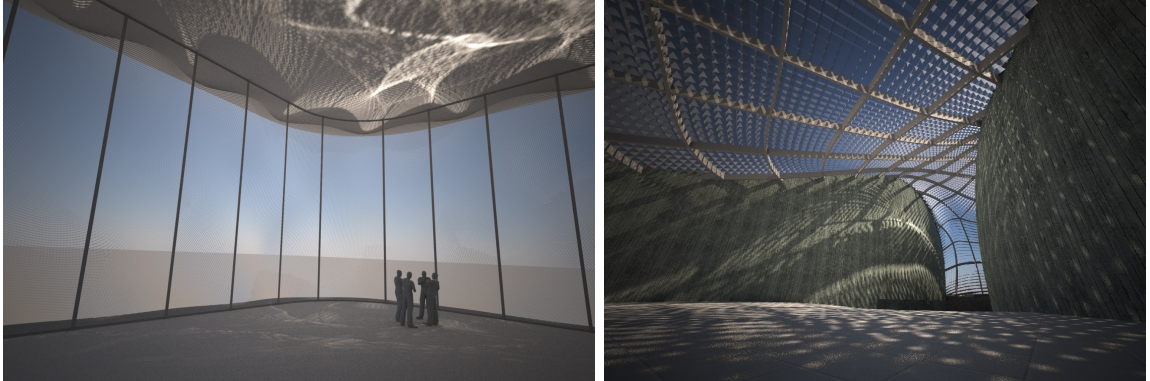


Figure 1.1: Line congruences constitute a basic geometric object in the computation of freeform shading and lighting systems for architecture. Our objects of study are discrete 2D systems of straight lines which undergo optimization according to geometric requirements, followed by conversion to a non-manifold quad mesh with planar faces which is capable of blocking light, creating reflection patterns, or serving as part of the structure.

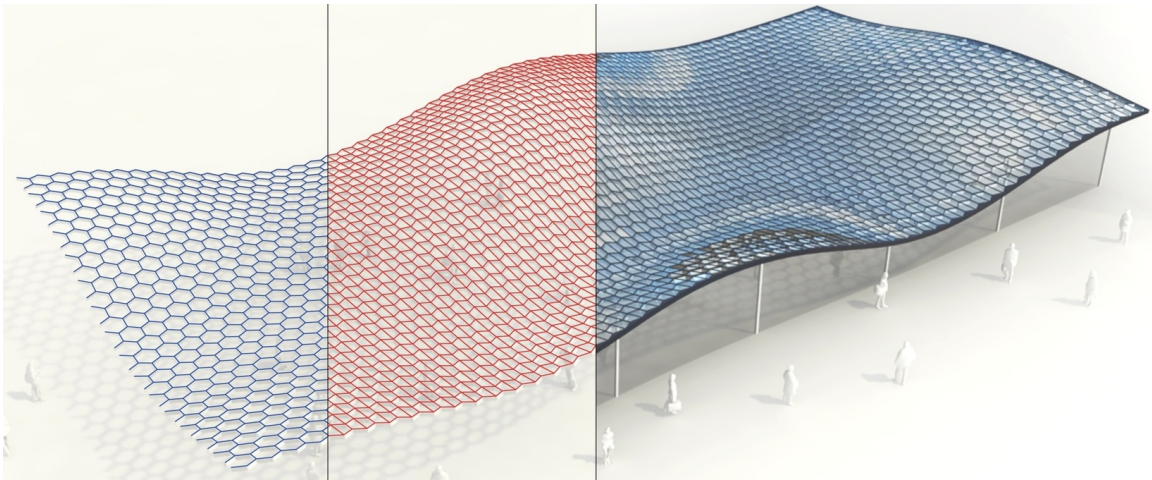


Figure 1.2: We approximate the Cour Visconti roof in the Louvre, Paris, by a quad mesh with planar faces which caps a honeycomb (consisting of hexagonal cells whose walls intersect at  $120^\circ$ ). This structure exhibits several features important in freeform architectural design: planar faces, low valence of nodes, a torsion-free support structure, and repetitive node geometry.

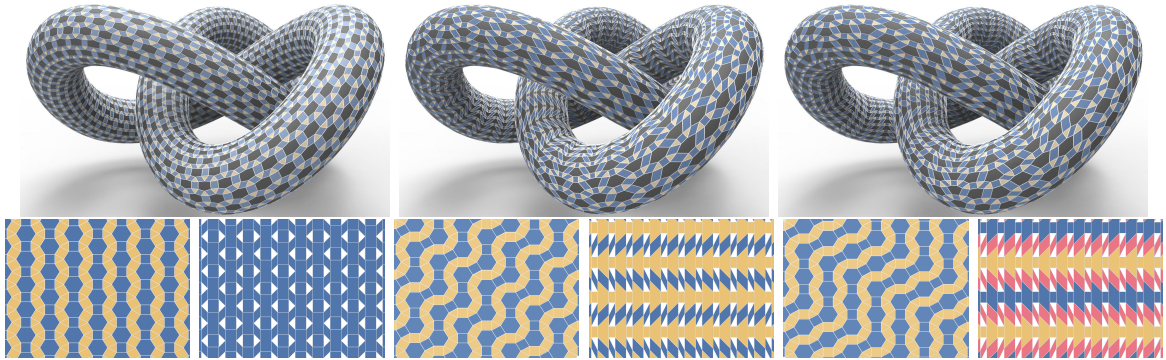


Figure 1.3: Polyhedral patterns on a knot. Top: three polyhedral patterns tiling a knot and optimized by our framework. All examples are combinatorially equivalent to a semi-regular pattern  $(3, 4, 6, 4)$ . Bottom: each of the three solutions is induced by a choice of *strip decomposition* and corresponding *affine symmetries*. For each model, we show the strip decomposition (left) with the pattern in the plane colored by yellow and blue strips. We show the deformed pattern upon mapping to a cylinder, suggesting the feasible symmetries (right). The different colors encode different choices of symmetries. For instance, blue faces are symmetric with respect to the barycenter.

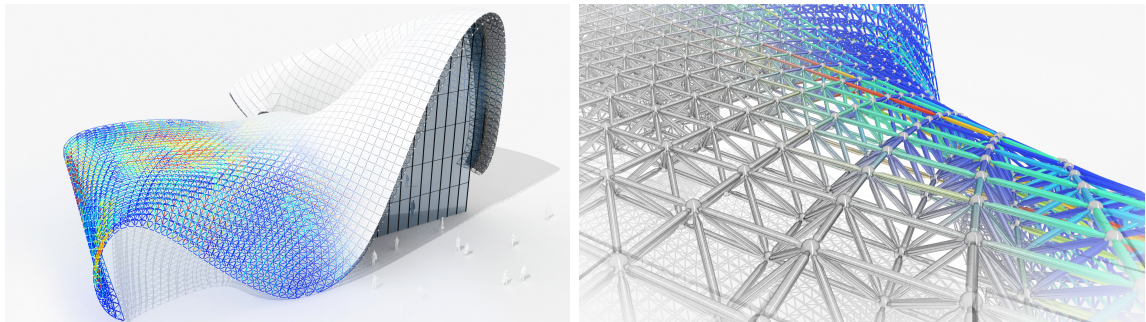


Figure 1.4: Left: A statically sound space structure designed and optimized with our framework, motivated by the real architectural project shown in Figure 5.1. Right: The space structure is constructed with six types of customized beams to minimize the total volume of the material used for beams while maintaining moderate manufacturing complexity. Here, a hotter color indicates a larger beam cross-section area. Our framework automatically determines the optimal cross-section areas of the six types of beams as well as the assignment of beam types.

# Chapter 2

## Shading and lighting systems

### 2.1 Introduction

This chapter studies objects which implicitly are important parts of graphics and geometry processing in several ways, namely *line congruences*. These are 2-parameter families of straight lines, of which the light rays emanating from a point source are an example. Another example of much richer geometry is the system of lines which intersect a surface orthogonally, and which is closely tied to the curvature behaviour of that surface [3]. The geometry of smooth line congruences and their many relations to surfaces are well understood. However there are only very few contributions to the topic of the present chapter, which is discrete line congruences and their relations to discrete surfaces. We here propose a discrete version of line congruences and discuss their fundamentals as well as applications in shading and lighting systems in architecture.

**Contributions.** These include discrete line congruences based on triangle meshes (§2.2) and the important case of discrete normal congruences (§2.3). Hitherto discrete congruences have mostly been studied in the form of their quad-based *torsal* parametrizations which could be interpreted as special quad-remeshings of triangle-based congruences. Both kinds are relevant for our work, since our main application – shading systems – essentially is the same as a torsal parametrization. Our algorithmic



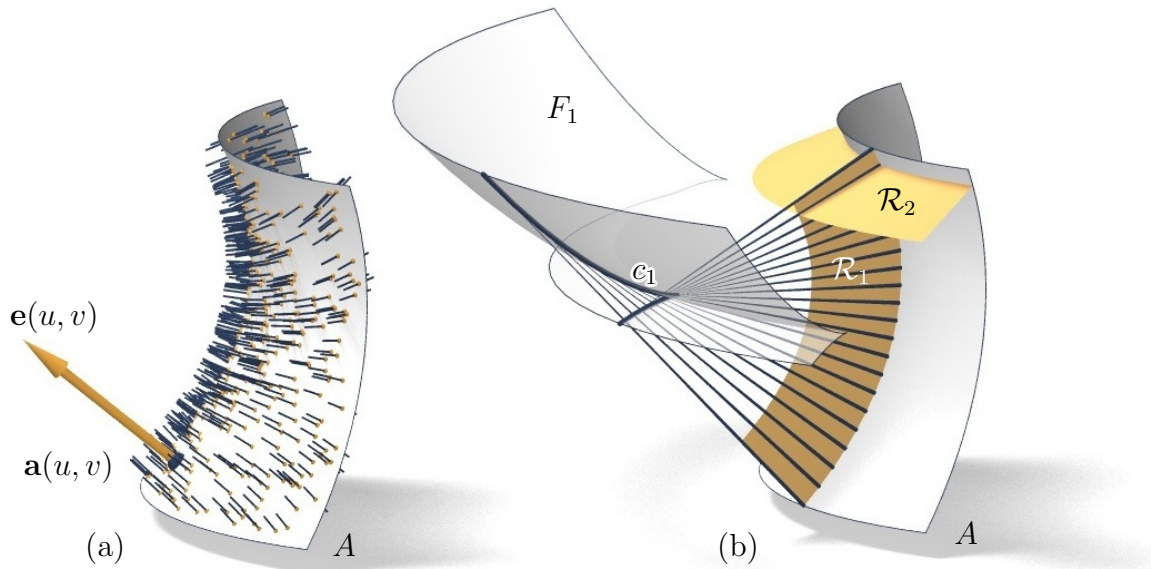


Figure 2.1: (a) A congruence  $\mathcal{L}$  of lines  $L(u, v)$  is described by a surface  $A$ , parametrized by  $\mathbf{a}(u, v)$ , and direction vectors  $\mathbf{e}(u, v)$ . (b) Developables  $\mathcal{R}_1, \mathcal{R}_2$  contained in  $\mathcal{L}$ . The set of all regression curves  $c_i$  of these developables makes up the focal sheets  $F_1, F_2$  of the congruence (here only  $F_1$  is shown).

contribution is a 2-stage optimization procedure: We optimize a triangle-based congruence (§2.4.1) such that quad-remeshing (§2.4.2) yields the desired shading system, up to a bit of final optimization. Results (§2.4.3) and discussion (§2.5) conclude this chapter.

**Previous work.** For an overview on line congruences with an emphasis on computing we refer to [4]. Design of congruences (with applications in mechanical engineering) has been studied by [5], who consider Bézier surfaces in an appropriate space of lines, thus modeling smooth congruences via a discrete control structure. Discrete normal congruences, with a computational framework for estimating focal surfaces of meshes with known or estimated normals, have been presented by [6]. There are some contributions to discrete line congruences in connection with special quad meshes (integrable systems), for which we refer to the monograph by [7]. They do not consider discrete versions of congruences, but rather discrete versions of the torsal parametrizations of congruences, which are also an important topic in the

present chapter. This theory has been first elaborated by [8]. Of particular interest is the special case of discrete normal congruences, which lead to torsion-free support structures in architectural geometry, special cases of which have already been considered in connection with meshes with specific offset properties [9–11]. We should mention that also semidiscrete versions of these constructions are of importance in architecture [12]. Finally refer to [13] and the references therein for light control in architecture.

## 2.2 Line Congruences

### 2.2.1 Smooth line congruences

We here recall a few facts from differential geometry. A line congruence  $\mathcal{L}$  is a smooth 2D manifold of lines described locally by lines  $L(u, v)$  which connect corresponding points  $\mathbf{a}(u, v)$  and  $\mathbf{b}(u, v)$  of two surfaces  $A, B$ . The vector  $\mathbf{e}(u, v) = \mathbf{b}(u, v) - \mathbf{a}(u, v)$  indicates the direction of the line  $L(u, v)$ .  $\mathcal{L}$  is equivalently described by the volume parametrization

$$\mathbf{x}(u, v, \lambda) = \mathbf{a}(u, v) + \lambda \mathbf{e}(u, v) = (1 - \lambda) \mathbf{a}(u, v) + \lambda \mathbf{b}(u, v).$$

A ruled surface  $\mathcal{R} \subset \mathcal{L}$  is described by functions  $u(t), v(t)$ : A parametrization of such a ruled surface via parameters  $t, \lambda$  is given by  $\mathbf{x}(u(t), v(t), \lambda)$ .

**Torsal directions.** In view of our applications we are especially interested in the *developable* ruled surfaces  $\mathcal{R}$  contained in the congruence  $\mathcal{L}$ . Using subscripts for partial derivatives, and the symbol  $[\cdot, \cdot, \cdot]$  for the determinant, the condition that  $\mathcal{R}$  is developable reads  $[\mathbf{e}, \mathbf{b}_t, \mathbf{a}_t] = 0$  or equivalently  $[\mathbf{e}, \mathbf{e}_t, \mathbf{a}_t] = 0$  [4]. It expands to

$$u_t^2 \underbrace{[\mathbf{e}_u, \mathbf{a}_u, \mathbf{e}]}_{=: \gamma_0(u,v)} + u_t v_t \underbrace{([\mathbf{e}_u, \mathbf{a}_v, \mathbf{e}] + [\mathbf{e}_v, \mathbf{a}_u, \mathbf{e}])}_{=: 2\gamma_1(u,v)} + v_t^2 \underbrace{[\mathbf{e}_v, \mathbf{a}_v, \mathbf{e}]}_{=: \gamma_2(u,v)} = 0. \quad (2.1)$$

For any fixed line  $L(u, v)$  of the congruence, Equation (2.1) has up to 2 solutions  $u_t : v_t$ , which are called *torsal directions*. By integrating torsal directions one creates functions  $u(t)$ ,  $v(t)$  which fulfill (2.1) and which describe developable surfaces  $\mathcal{R}$  contained in  $\mathcal{L}$  (see Fig. 2.1).

**Example: Normal Congruences** (see also §2.3). A classical example of a congruence is formed by the lines orthogonal to a surface  $A$ . In this case the normals along a principal curvature line constitute a developable ruled surface [3]. Thus such *normal congruences* always have torsal directions, namely the principal directions of  $A$  (see Fig. 2.1).

**REMARK** (*Undefined torsal directions*). In the special case that  $\mathcal{L}$  consists of the bundle of lines incident with a center, all ruled surfaces  $\mathcal{R} \subset \mathcal{L}$  are cones (and thus developable) and all directions are torsal. It is important for us to know that such cases can occur, since optimization of congruences later in this chapter may yield congruences close to a bundle, and defining a smooth frame field of torsal directions has to be assisted e.g. by a smoothness energy.

**Focal points.** We are especially interested in *hyperbolic* congruences where two torsal directions exist everywhere. Any line  $L(u, v) \in \mathcal{L}$  is then contained in two developables  $\mathcal{R}_1, \mathcal{R}_2$ . It is known that this happens if and only if

$$([\mathbf{a}_u, \mathbf{e}_v, \mathbf{e}] + [\mathbf{e}_u, \mathbf{a}_v, \mathbf{e}])^2 \geq 4[\mathbf{e}_u, \mathbf{e}_v, \mathbf{e}][\mathbf{a}_u, \mathbf{a}_v, \mathbf{e}]. \quad (2.2)$$

To understand (2.2) we observe that among ruled surfaces, developables are charac-

terized by having singular points on otherwise regular rulings (the curves of regression of Fig. 2.1). Thus, hyperbolicity implies that there exist singularities (*focal points*)  $\mathbf{x}(u, v, \lambda)$ , where  $[\mathbf{x}_u, \mathbf{x}_v, \mathbf{x}_\lambda] = 0$ , i.e.,

$$[\mathbf{e}_u, \mathbf{e}_v, \mathbf{e}] \lambda^2 + ([\mathbf{a}_u, \mathbf{e}_v, \mathbf{e}] + [\mathbf{e}_u, \mathbf{a}_v, \mathbf{e}]) \lambda + [\mathbf{a}_u, \mathbf{a}_v, \mathbf{e}] = 0. \quad (2.3)$$

(2.3). This equation has solutions if and only if its discriminant is nonnegative, whence (2.2). For the converse statement and “singularities at infinity” we refer to [4]. Summing up, (2.2) holds  $\iff$  (2.3) has solutions  $\iff$  (2.1) has solutions.

**Example: Congruences defined by affine mappings.** Congruences defined by parametrizations of the form

$$\mathbf{x}(u, v, \lambda) = \mathbf{a}_0 + \mathbf{a}_{10}u + \mathbf{a}_{20}v + \lambda(\mathbf{e}_0 + \mathbf{e}_{10}u + \mathbf{e}_{20}v)$$

play an important in this work. For fixed  $\lambda = \lambda_0$ , the mapping  $\mathbf{x}(u, v, \lambda_0)$  parametrizes a plane  $P_{\lambda_0}$ . The affine mapping from  $P_\alpha$  to  $P_\beta$ ,

$$\mathbf{x}(u, v, \alpha) \longmapsto \mathbf{x}(u, v, \beta),$$

connects points which span the lines of the congruence. Focal points can be computed by (2.3). The following properties (see [4, Ex. 7.1.2] and Fig. 2.2) are important for us:

1. Each line  $L = P_\alpha \cap P_\beta$  is contained in  $\mathcal{L}$
2. The lines  $P_\alpha \cap P_\beta$  with  $\alpha$  fixed, constitute a developable surface  $\mathcal{R} \subset \mathcal{L}$  which is planar and contained in  $P_\alpha$  (in general, it is the tangent surface of a parabola).

We are going to make use of these congruences in the next subsection, when we consider *discrete* congruences defined by a correspondence between triangle meshes.

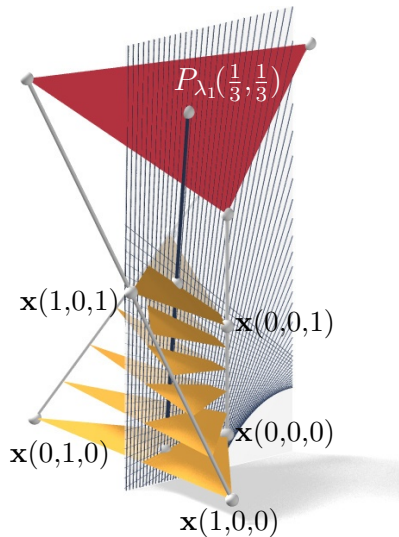


Figure 2.2: Congruence  $\mathcal{L}$  defined by a “linear” volumetric parametrization  $\mathbf{x}(u, v, \lambda)$ . Planes  $P_\lambda$  defined by  $\lambda = \text{const}$  are visualized as triangles. The red triangle  $P_{\lambda_1}$  contains the ruling  $L(\frac{1}{3}, \frac{1}{3})$  and so the set of lines  $P_{\lambda_1} \cap P_\beta$ ,  $\beta \in \mathbb{R}$ , constitutes a developable  $\mathcal{R} \subset \mathcal{L}$  through that ruling.

### 2.2.2 Congruences defined over triangle meshes

Let us define discrete congruences by means of two combinatorially equivalent triangle meshes  $A, B$  with vertices  $\{\mathbf{a}_i\}$  and  $\{\mathbf{b}_i\}$ . The correspondence  $\mathbf{a}_i \longleftrightarrow \mathbf{b}_i$  defines, via linear interpolation, correspondences between corresponding faces  $\mathbf{a}_i \mathbf{a}_j \mathbf{a}_k$  and  $\mathbf{b}_i \mathbf{b}_j \mathbf{b}_k$ . Connecting corresponding points then yields a congruence  $\mathcal{L}$  which is composed of pieces of the congruences studied in the example above. For each pair of corresponding triangles we let

$$\mathbf{e}_i = \mathbf{b}_i - \mathbf{a}_i, \quad \mathbf{a}_{ij} = \mathbf{a}_i - \mathbf{a}_j, \quad \mathbf{e}_{ij} = \mathbf{e}_i - \mathbf{e}_j$$

and obtain a volumetric parametrization of the type described in the example above:

$$\begin{aligned} \mathbf{x}(u, v, \lambda) &= \mathbf{a}(u, v) + \lambda \mathbf{e}(u, v), \\ \mathbf{a}(u, v) &= \mathbf{a}_i + u \mathbf{a}_{ji} + v \mathbf{a}_{ki}, \quad \mathbf{e}(u, v) = \mathbf{e}_i + u \mathbf{e}_{ji} + v \mathbf{e}_{ki}. \end{aligned} \tag{2.4}$$

Here  $u, v$  run in the triangular domain  $u, v, 1 - u - v \geq 0$ . In each point of a triangle we may now use Equations (2.1) and (2.3) to compute torsal directions and focal points.

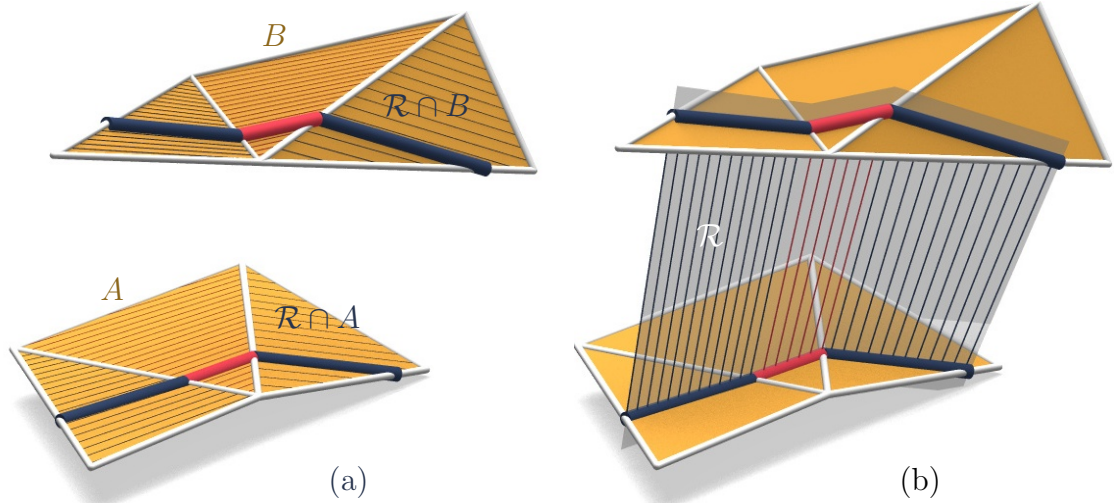


Figure 2.3: Piecewise-linear correspondence of meshes  $A$ ,  $B$  defining a piecewise-smooth congruence  $\mathcal{L}$ . (a) Integrating torsal directions yields corresponding polylines  $A$  and  $B$ . (b) Connecting corresponding points of those polylines yields a piecewise-flat (and thus developable) surface  $\mathcal{R} \subset \mathcal{L}$ .

In order to get a developable  $\mathcal{R}$  contained in the congruence  $\mathcal{L}$ , we pick an initial point in a face and integrate the torsal directions from there. Property 2 above shows that as long as we stay within a face, the torsal directions integrate along straight lines (see Figure 2.3). When stepping over an edge from one triangle into the next one there are up to two possible torsal directions to continue, and we choose the one which has minimal deviation from the previous one. This procedure yields polylines  $\mathcal{R} \cap A$  and  $\mathcal{R} \cap B$ , which subsequently span the developable  $\mathcal{R}$ , and which are considered a discrete representation of the developable  $\mathcal{R}$ .

### 2.2.3 Congruences defined over quad meshes

Algorithms on meshes frequently have the aim that their results depend as little as possible on the meshing but rather on geometric properties of the underlying assumed smooth shape which is approximated by the mesh. Sometimes however the mesh has regular combinatorics and the actual mesh polylines play an important role. This section, changing from triangle-based congruences to quad-based ones, performs exactly this change of viewpoint. Similar to §2.2.2, assume combinatorially

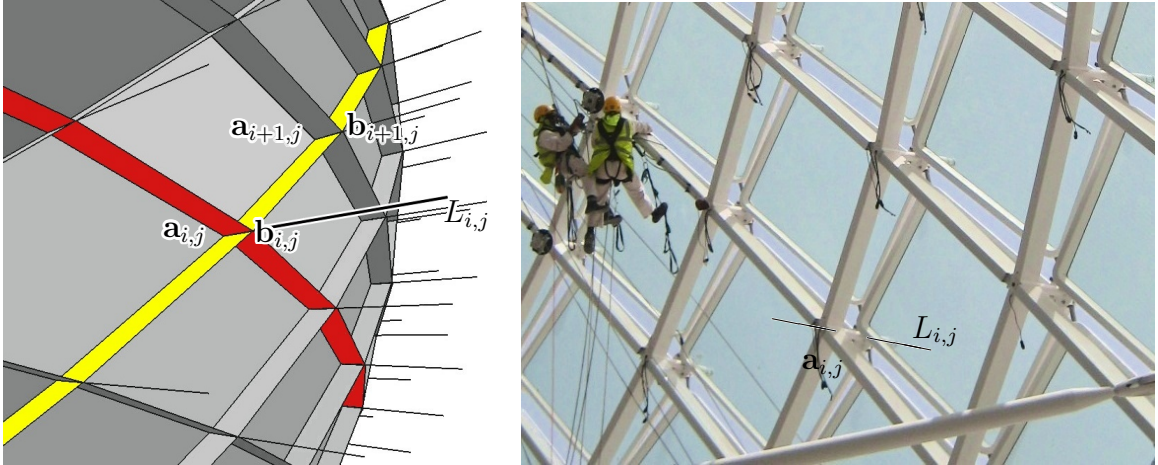


Figure 2.4: A torsal discrete congruence  $\mathcal{L}$  is defined by connecting corresponding vertices of quad meshes  $A, B$  where corresponding edges are co-planar; creating discrete developables (red and yellow) along mesh polylines. This congruence has been used for the Yas Marina hotel, Abu Dhabi (right) to create a torsion-free support structure.

equivalent quad meshes  $A, B$ . We could imitate the construction of §2.2.2 and define a piecewise-smooth congruence by bilinear interpolation within faces, but this does not lead to new insights. We therefore do not pursue this direction and reserve quad combinatorics for the treatment of *torsal parametrizations* of congruences:

**DEFINITION 1.** *A parametrization  $L(u, v)$  of a congruence is torsal, if the ruled surfaces defined by  $u = \text{const}$  are developable, and so are the ones defined by  $v = \text{const}$*

A discrete torsal parametrization is defined by a direct analogy: A discrete ruled surface (a sequence of lines) is developable if successive lines are co-planar (see colored ruled surfaces in Figure 2.4). Using this notion, we define:

**DEFINITION 2.** *A line congruence  $\{L_{i,j}\}$  of regular quad combinatorics is a torsal parametrization, if the ruled surfaces defined by  $i = \text{const}$  are developable, and so are those defined by  $j = \text{const}$*

**Application: Torsion-Free Support Structures.** Figure 2.4 shows a steel construction where prismatic beams follow the edges of a mesh, denoted by  $A$ , with regular quad combinatorics.  $A$  has the additional property that each vertex  $\mathbf{a}_{i,j}$  is

equipped with a straight line  $L_{i,j}$  such that for each beam adjacent to that vertex, the central symmetry plane of the beam contains  $L_{i,j}$ . Obviously this happens if and only if the lines  $L_{i,j}$  constitute a discrete torsal parametrization.

Such constructions play an important part in the geometry of freeform architecture. We give them the name under which they are usually referred to in this context:

**DEFINITION 3.** *A torsion-free support structure consists of combinatorially equivalent meshes  $A, B$  such that corresponding edges are co-planar but do not coincide (in case of regular quad combinatorics, lines connecting corresponding vertices of  $A, B$  constitute a torsal parametrization).*

Doliwa et al. [8], who first studied discrete torsal parametrizations in depth, use the word *conjugacy* for the relation between the mesh  $A$  and the congruence  $\mathcal{L}$ . Previous work on torsion-free support structures was in the context of meshes with planar faces: [10] treat support structures in the context of architectural geometry. Actually planarity of faces of  $A$  is an unnecessary restriction, see Fig. 2.4. It is an aim of the present chapter to study and compute support structures consisting of meshes  $A, B$  with non-planar faces, and to use them for new purposes.

## 2.3 Normal congruences

### 2.3.1 Smooth normal congruences

We already mentioned *normal congruences*, which are formed by the surface normals of a smooth surface  $A$  (see Figure 2.1). The volume parametrization corresponding to such a congruence  $\mathcal{L}$  reads  $\mathbf{x}(u, v, \lambda) = \mathbf{a}(u, v) + \lambda \mathbf{e}(u, v)$ , where  $\mathbf{a}(u, v)$  parametrizes the surface  $A$ , and  $\mathbf{e}(u, v)$  is the unit normal vector field. Note that any constant-distance offset  $A^d$  of  $A$  defines the same congruence, with  $\mathbf{a}^d = \mathbf{a} + d\mathbf{e}$ ,  $\mathbf{e}^d = \mathbf{e}$ , and  $\mathbf{x}^d(u, v, \lambda) = \mathbf{x}(u, v, \lambda + d)$ .



REMARK (*Relation to Surfaces*). Properties of normal congruences correspond directly to properties of surfaces: Torsal directions of  $\mathcal{L}$  correspond to principal directions of  $A$  ( $\implies$  torsal directions exist everywhere). A developable surface in  $\mathcal{L}$  consists of the surface normals along a principal curvature line of  $A$  ( $\implies$  there are two families of developables which intersect at right angles; actually this characterizes normal congruences). The focal surfaces of  $\mathcal{L}$  consist of principal curvature centers of  $A$ , so they are a surface analogue of the evolute of a curve [3].

A general congruence  $\mathbf{x}(u, v, \lambda) = \mathbf{a}(u, v) + \lambda \mathbf{e}(u, v)$  might be the normal congruence of an as yet unknown surface  $\mathbf{a}^*(u, v)$  with normal vectors  $\mathbf{e}(u, v)$ . In order to find out if this is the case, we write  $\mathbf{a}^*(u, v) = \mathbf{a}(u, v) + \lambda(u, v)\mathbf{e}(u, v)$  and solve for  $\lambda(u, v)$ . If we restrict ourselves to  $\|\mathbf{e}\| = 1$  we get  $\langle \mathbf{e}, \mathbf{e}_u \rangle = \langle \mathbf{e}, \mathbf{e}_v \rangle = 0$ , and the orthogonality conditions  $\langle \mathbf{e}, \mathbf{a}_u^* \rangle = \langle \mathbf{e}, \mathbf{a}_v^* \rangle = 0$  are equivalent to  $\lambda_u = -\langle \mathbf{a}_u, \mathbf{e} \rangle$ ,  $\lambda_v = -\langle \mathbf{a}_v, \mathbf{e} \rangle$ . This PDE has a solution if and only if the integrability condition  $\lambda_{uv} = \lambda_{vu}$  holds, i.e.,

$$\langle \mathbf{a}_u, \mathbf{e}_v \rangle = \langle \mathbf{a}_v, \mathbf{e}_u \rangle. \quad (2.5)$$

### 2.3.2 Discrete normal congruences

The following definition takes up a property characterizing smooth normal congruences:

DEFINITION 4. *A congruence  $\mathcal{L}$  defined by a piecewise-linear correspondence of triangle meshes  $A, B$  is called normal, if torsal planes (spanned by ruling and torsal direction) in the barycenters of faces are orthogonal.*

We consider corresponding faces  $\mathbf{a}_1\mathbf{a}_2\mathbf{a}_3$  and  $\mathbf{b}_1\mathbf{b}_2\mathbf{b}_3$  of  $A, B$ , with difference vectors  $\mathbf{e}_i = \mathbf{b}_i - \mathbf{a}_i$  and project them onto a plane orthogonal to the line connecting their barycenters. This yields triangles  $\bar{\mathbf{a}}_1\bar{\mathbf{a}}_2\bar{\mathbf{a}}_3$  and  $\bar{\mathbf{b}}_1\bar{\mathbf{b}}_2\bar{\mathbf{b}}_3$  and vectors  $\bar{\mathbf{e}}_i = \bar{\mathbf{b}}_i - \bar{\mathbf{a}}_i$  (see Figure 2.5). A discrete analogue of (2.5) now is the following:

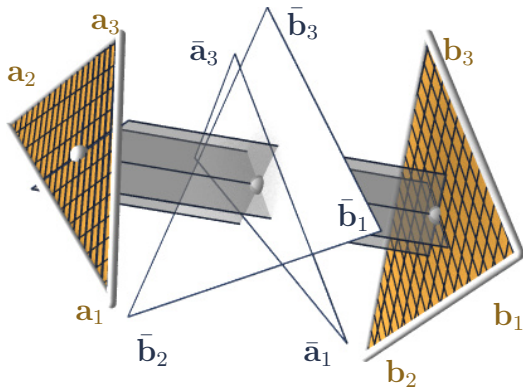


Figure 2.5: Congruences defined by corresponding meshes  $A, B$  are *normal* if torsal planes in the barycenters of faces are orthogonal (here we show also the projection used by Prop. 1).

PROPOSITION 1. *In the notation of the previous paragraph, meshes  $A, B$  define a normal congruence  $\iff$  for each pair of corresponding faces, we have*

$$\langle \bar{\mathbf{a}}_{ij}, \bar{\mathbf{b}}_{ik} \rangle = \langle \bar{\mathbf{a}}_{ik}, \bar{\mathbf{b}}_{ij} \rangle, \quad (2.6)$$

where  $\bar{\mathbf{a}}_{ij} = \bar{\mathbf{a}}_j - \bar{\mathbf{a}}_i$ ,  $\bar{\mathbf{b}}_{ij} = \bar{\mathbf{b}}_j - \bar{\mathbf{b}}_i$ . This is equivalent to

$$\langle \bar{\mathbf{a}}_{ij}, \bar{\mathbf{e}}_{ik} \rangle = \langle \bar{\mathbf{a}}_{ik}, \bar{\mathbf{e}}_{ij} \rangle,$$

where  $\bar{\mathbf{e}}_{ij} = \bar{\mathbf{e}}_j - \bar{\mathbf{e}}_i$ . It is sufficient that these equations hold for at least one choice of indices  $i \neq j \neq k$ .

*Proof.* Corresponding points  $\mathbf{a} \in A$ ,  $\mathbf{b} \in B$  move in corresponding torsal directions  $\mathbf{a}_t$ ,  $\mathbf{b}_t$ , resp., if and only if  $\mathbf{b} - \mathbf{a}$ ,  $\mathbf{a}_t$ ,  $\mathbf{b}_t$  are coplanar, cf. the text above (2.1). With  $\mathbf{a}, \mathbf{b}$  as barycenters of corresponding faces, this is obviously equivalent to linear dependence of  $\bar{\mathbf{a}}_t, \bar{\mathbf{b}}_t$ . When using the projection  $\bar{\mathbf{a}} = \bar{\mathbf{b}}$  of barycenters as the origin of the coordinate system, there is a linear mapping  $\alpha$  which maps corresponding points  $\bar{\mathbf{a}}_i \mapsto \bar{\mathbf{b}}_i$  ( $i = 1, 2, 3$ ) as well as vectors  $\bar{\mathbf{a}}_t \mapsto \bar{\mathbf{b}}_t$  (which are thus seen as eigenvectors of  $\alpha$ ). This implies that normality is characterized by orthogonality of  $\alpha$ 's eigenvectors, i.e., symmetry  $\langle \mathbf{x}, \alpha(\mathbf{y}) \rangle = \langle \alpha(\mathbf{x}), \mathbf{y} \rangle$  for at least 1 pair of linearly independent vectors  $\mathbf{x}, \mathbf{y}$ . This is exactly what is stated.  $\square$

REMARK. It is easy to find conditions equivalent to (2.6). The following ones involve

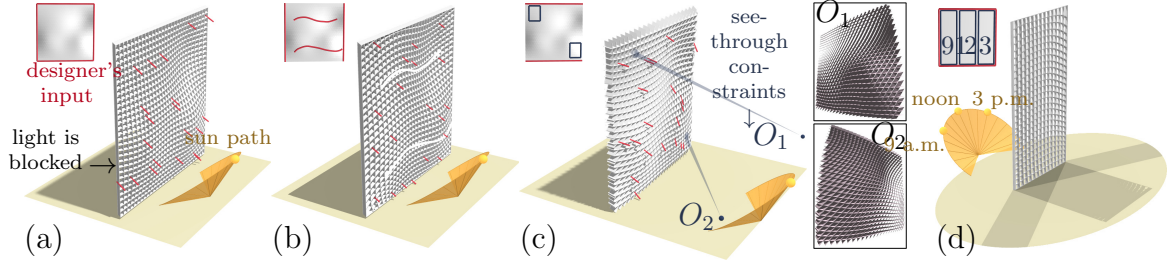


Figure 2.6: Shading systems with multiple constraints, computed by optimizing a line congruence (selected lines shown in red), subsequent conversion to torsal form and optimization towards planarity of shading fins. (a) Light is to be blocked, and the boundary, torsal directions are to be aligned with the boundary. (b) Light is to be blocked, torsal directions are to be aligned with a user's design strokes. (c) Light is to be blocked, and in two selected areas of the facade, specified objects are to be visible (see inset figures at right for fish-eye views from  $O_1$  and  $O_2$  which verify this see-through constraint). (d) Here a truly flat facade is equipped with a shading system whose different parts block light emitted from different sun positions.

the difference of face centers,

$$\mathbf{e}_c = \frac{1}{3}(\mathbf{b}_1 + \mathbf{b}_2 + \mathbf{b}_3) - \frac{1}{3}(\mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3),$$

which indicates the direction of projection. We have

$$\begin{aligned} (2.6) \iff \langle \mathbf{a}_{ij} \times \mathbf{e}_c, \mathbf{b}_{ik} \times \mathbf{e}_c \rangle &= \langle \mathbf{a}_{ik} \times \mathbf{e}_c, \mathbf{b}_{ij} \times \mathbf{e}_c \rangle \\ \iff \langle \mathbf{a}_{ij} \times \mathbf{e}_c, \mathbf{e}_{ik} \times \mathbf{e}_c \rangle &= \langle \mathbf{a}_{ik} \times \mathbf{e}_c, \mathbf{e}_{ij} \times \mathbf{e}_c \rangle. \end{aligned} \quad (2.7)$$

## 2.4 Applications and Algorithms

The importance of torsion-free support structures for steel constructions has already been emphasized, see Fig. 2.4 and [10]. We therefore demonstrate the capabilities of modeling with line congruences by means of another application, namely *freeform shading and lighting systems*.

Torsion-free support structures, which exhibit many planar quads, are well suited to function as shading elements themselves — see Figure 2.6. Their design is based

on optimization of a line congruence  $\mathcal{L}$ , and subsequent extraction of a torsion-free support structure from  $\mathcal{L}$  whose planes (i.e., torsal planes of  $\mathcal{L}$ ) have the function of blocking light. It is very important that the *combinatorics* of the shading system is determined only in the second step, after optimization of the congruence has been performed.

A typical design objective for shading systems applications is the blocking of light by shading fins which correspond to torsal planes. We could require

- (A). One family of  $\mathcal{L}$ 's torsal planes is as orthogonal to incoming light as possible (so that those planes can function as shading fins of minimal possible width).
- (B). As an alternative, the lines of  $\mathcal{L}$  are as orthogonal as possible to the incoming light rays.

(A) achieves the goal of blocking light in a more obvious manner than (B), which indiscriminately moves all quads in the support structure in a position generally transverse to the light rays. Numerical experiments suggest that requiring (B) has the same effect as (A). Since it is simpler to implement we therefore employed (B) in most examples. The foundation of this observation is

**PROPOSITION 2.** *For parallel light, generically (B)  $\implies$  (A).*

*Proof.* (B)  $\implies$  rulings of developables  $\mathcal{R} \subset \mathcal{L}$  are orthogonal to light. Such developables can only be cylinders or planar. Generically not both families of developables in  $\mathcal{L}$  are cylinders since then all rulings would be parallel. So at least one family of developables is planar, and torsal planes (being tangent planes of  $\mathcal{R}$ ) are orthogonal to light rays. □

This proof shows that congruences fulfilling (A) or (B) in an exact manner are rather special and will not occur in practice. Accordingly our examples achieve (A) or (B) only in a least squares sense. Further design objectives are:

- (C). The user might prescribe individual lines of  $\mathcal{L}$ . Fitting a congruence to these data is not difficult [4], but in view of applications we want to do it such that  $\mathcal{L}$  is hyperbolic and convertible to a torsal parametrization.
- (D). Hyperbolicity is achieved by incorporating (2.6) into the optimization, making  $\mathcal{L}$  more “normal”.
- (E). A user may prescribe torsal directions at selected locations of the mesh, in order to guide the appearance of the support structure later extracted from  $\mathcal{L}$ .
- (F). A similar design objective is that incoming light is reflected in torsal planes in a prescribed way.

### 2.4.1 Optimization of Mesh-based Congruences

Most of the design objectives formulated above involve global optimization, and the following paragraphs show how to do that. We discuss how to optimize a congruence  $\mathcal{L}$  defined by a fixed triangle mesh  $A = (V, E, F)$  and a variable triangle mesh  $B$  (we keep  $A$  fixed since we are later remeshing anyway). Throughout this text, corresponding vertices of meshes  $A, B$  are given by

$$\mathbf{a}_i \quad \text{and} \quad \mathbf{b}_i = \mathbf{a}_i + \mathbf{e}_i \quad \text{with} \quad \|\mathbf{e}_i\| \approx 1.$$

The lines  $L_i$  of the congruence connect vertices  $\mathbf{a}_i$  and  $\mathbf{b}_i$ . Restriction to unit vectors  $\mathbf{e}_i$  yields simpler expressions for target functionals, at the cost of some degrees of freedom.

**Contributions to target functionals.** Below we list the components used to build the various target functionals for optimization which are employed in individual examples. Using an average edge length  $\delta$ , we appropriately normalize each term in order to make it scale invariant.

- *Fairness.* Assuming that  $A$  is fair, we express fairness of the congruence in terms of

the Laplacian of vectors  $\mathbf{e}_i$  interpreted as a vector-valued function “ $\mathbf{e}$ ” on the mesh  $A$ :

$$f_{\text{fair}} = \frac{1}{|V|} \sum_{\mathbf{a}_i \in V} \|\Delta \mathbf{e}_i\|^2, \quad \text{where } \Delta \mathbf{e}_i = \mathbf{e}_i - \frac{1}{\deg \mathbf{a}_i} \sum_{\mathbf{a}_j \sim \mathbf{a}_i} \mathbf{e}_j.$$

- *The normal congruence property.* In the notation of (2.7), we penalize deviation from that property by

$$f_{\text{norm}} = \frac{1}{\delta^2 |F|} \sum_{\mathbf{a}_i \mathbf{a}_j \mathbf{a}_k \in F} (\langle \mathbf{a}_{ij} \times \mathbf{e}_c, \mathbf{e}_{ik} \times \mathbf{e}_c \rangle - \langle \mathbf{a}_{ik} \times \mathbf{e}_c, \mathbf{e}_{ij} \times \mathbf{e}_c \rangle)^2$$

- *Hyperbolicity constraint.* If our congruence is to have torsal planes, the discriminant condition (2.2) must hold everywhere. For practical purposes we require it for the barycenters of each face  $\Delta = \mathbf{a}_i \mathbf{a}_j \mathbf{a}_k$ . Using (2.4), it expands to

$$\begin{aligned} c_{\text{hyp}}(\Delta) &= ([\mathbf{a}_{ij}, \mathbf{e}_{ik}, \mathbf{e}_c] + [\mathbf{e}_{ij}, \mathbf{a}_{ik}, \mathbf{e}_c])^2 \\ &\quad - 4[\mathbf{e}_{ij}, \mathbf{e}_{ik}, \mathbf{e}_c][\mathbf{a}_{ij}, \mathbf{a}_{ik}, \mathbf{e}_c] \geq 0. \end{aligned}$$

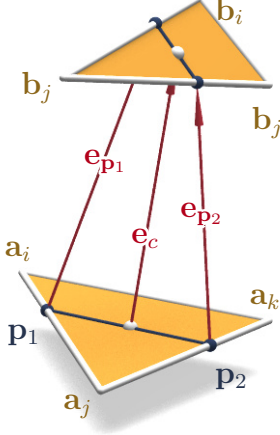
- *User-defined constraints.* If the user specifies that the line  $L_i \in \mathcal{L}$  should be parallel (resp., orthogonal) to a certain direction  $\mathbf{d}_i$ , we add appropriate linear combinations of

$$f_{\text{par},i} = \|\mathbf{e}_i \times \mathbf{d}_i\|^2, \quad \text{resp.,} \quad f_{\text{perp},i} = \langle \mathbf{e}_i, \mathbf{d}_i \rangle^2$$

to the target functional, depending on the application. The constraint that the angle between  $L_i$  and a user-specified vector  $\mathbf{d}_i$  does not exceed a certain threshold is expressed as  $\langle \mathbf{e}_i, \mathbf{d}_i \rangle - \text{const} \geq 0$  (here  $\|\mathbf{e}_i\| = 1$  is needed).

- *Prescribing torsal directions and planes.* If a user prescribes torsal directions in some part of the mesh, then we try to fulfill this wish for all faces  $\Delta = \mathbf{a}_i \mathbf{a}_j \mathbf{a}_k$  which

intersect that area of interest. We represent the required direction via a line segment  $\mathbf{p}_1\mathbf{p}_2 \subset \Delta$  containing the barycenter  $\mathbf{c} = \frac{\mathbf{a}_i + \mathbf{a}_j + \mathbf{a}_k}{3}$ . Fig. 2.3 makes it clear that  $\mathbf{p}_1\mathbf{p}_2$

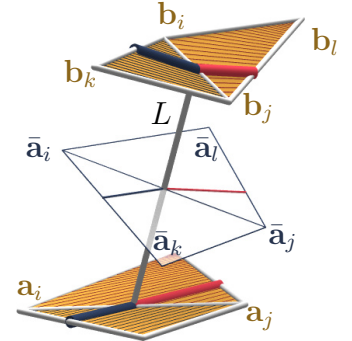


is torsal if and only if the lines of  $\mathcal{L}$  passing through the points  $\mathbf{p}_1, \mathbf{p}_2, \mathbf{c}$  (indicated by vectors  $\mathbf{e}_{\mathbf{p}_1}, \mathbf{e}_{\mathbf{p}_2}, \mathbf{e}_c$ ) are coplanar. A user wishing to prescribe an entire torsal plane must in addition specify its normal vector  $\mathbf{n}$  which in particular is then orthogonal to  $\mathbf{e}_c$ . Summing up, for penalizing deviation from a desired torsal direction and plane, we use

$$f_{\text{dir}}(\Delta) = [\mathbf{e}_{\mathbf{p}_1}, \mathbf{e}_{\mathbf{p}_2}, \mathbf{e}_c]^2, \quad f_{\text{plane}}(\Delta) = f_{\text{dir}}(\Delta) + \langle \mathbf{e}_c, \mathbf{n} \rangle^2.$$

- *Transversality of torsal planes.* For applications it is often desirable that torsal planes intersect at right angles or nearly so. If we are optimizing towards a normal congruence, this property is automatic. Otherwise we use a condition of the form  $c_{\text{ang}}(\Delta) > 0$  which holds true if and only if the angle between torsal directions in the face  $\Delta = \mathbf{a}_i\mathbf{a}_j\mathbf{a}_k$  does not fall below  $\alpha$  ( $c_{\text{ang}}$  is a function taking arguments  $\mathbf{a}_{ij}, \mathbf{a}_{ik}, \mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_k, \alpha$  and is not printed here).

- *Fairness of torsal directions* is expressed in the smallness of jump in torsal planes when crossing an edge. Consider the line  $L \in \mathcal{L}$  passing through the midpoint of an edge  $\mathbf{a}_i\mathbf{a}_j$  of the mesh  $A$  and project the adjacent triangles  $\mathbf{a}_i\mathbf{a}_j\mathbf{a}_k$  and  $\mathbf{a}_i\mathbf{a}_j\mathbf{a}_l$  orthogonally in direction  $L$ . This results in vertices  $\bar{\mathbf{a}}_i, \dots$



“No jump” is expressed by the condition that torsal directions in the adjacent triangles project onto the same straight line. The same procedure can be applied to the mesh  $B$ . It is not difficult to verify that “no jump” is equivalently expressed by  $\bar{\mathbf{a}}_l$  having the same barycentric coordinates w.r.t.  $\bar{\mathbf{a}}_i\bar{\mathbf{a}}_j\bar{\mathbf{a}}_k$  as  $\bar{\mathbf{b}}_l$  has w.r.t.  $\bar{\mathbf{b}}_i\bar{\mathbf{b}}_j\bar{\mathbf{b}}_k$ . An appropriate sum of squares constitutes a *fairness energy*  $f_{\text{fair}/t}$  and is added to the

target functional as a regularizer.

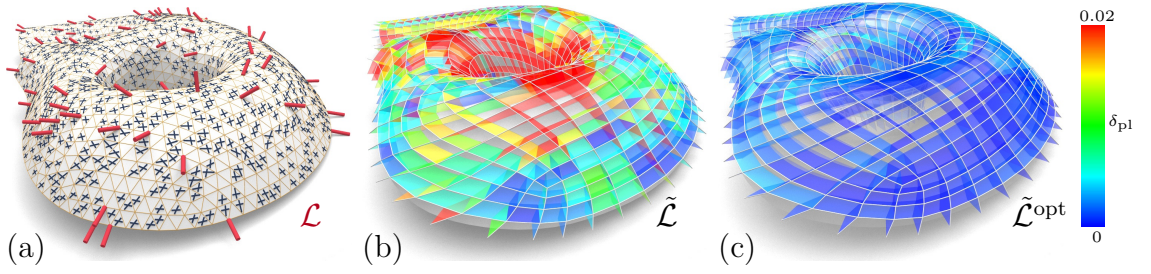


Figure 2.7: Flowchart of algorithm. (a) Optimize discrete line congruence  $\mathcal{L}$  (red) defined by meshes  $A, B$  (variables are vertices of  $A, B$ ) and compute torsal directions in the faces of  $A$  (blue). (b) Extract a quad-based support structure  $\tilde{\mathcal{L}}$  (an almost-torsal parametrization of  $\mathcal{L}$ ) which follows the torsal directions of  $\mathcal{L}$ . (c) Optimize the quads of  $\tilde{\mathcal{L}}$  for planarity.

**Unconstrained and constrained optimization.** We initialize optimization with vectors  $\mathbf{e}_i$  which are estimates for normal vectors in vertices of  $A$ . We employ both unconstrained and constrained optimization. In the unconstrained case we minimize a combination of  $f_{\text{fair}}$ ,  $f_{\text{norm}}$ , together with terms  $f_{\text{dir}}(\Delta)$ ,  $f_{\text{plane}}(\Delta)$  and other terms which correspond to design specifications. This optimization problem is solved by a quasi-Newton method (limited-memory BFGS method [14]). The constraint  $\|\mathbf{e}_i\| = 1$  is enforced by simply re-normalizing all  $\mathbf{e}_i$ 's after each round of iteration.

We also perform constrained optimization of the same kind of target functional, by adding user-defined constraints like  $c_{\text{hyp}}(\Delta) \geq 0$  or  $c_{\text{ang}}(\Delta) \geq 0$ . We employ an augmented Lagrangian method to solve this constrained optimization problem. Again  $\|\mathbf{e}_i\| = 1$  is enforced by re-normalization.

## 2.4.2 Conversion to Quad-based Torsal Form

Converting the congruence  $\mathcal{L}$  (defined by triangle meshes  $A, B$ ) to torsal form means finding a discrete torsal parametrization  $\tilde{\mathcal{L}}$  (defined by quad meshes  $\tilde{A}, \tilde{B}$ ) whose lines fit in the original congruence  $\mathcal{L}$ . The easiest method of conversion is to choose  $\tilde{A}, \tilde{B}$  as respective remeshings of  $A, B$ , because then the edges of  $\tilde{A}$  follow the torsal



directions of  $\mathcal{L}$  in  $A$ . The actual construction of  $\tilde{\mathcal{L}}$  requires the two steps *remeshing* and *optimization* (see Figure 2.7).

**Torsal Remeshing of Congruences.** Still using the notation from above, we first compute the frame field in  $A$  which indicates the torsal directions of the congruence  $\mathcal{L}$  (Fig. 2.7a). It is sufficient to compute the torsal directions by solving (2.1) for the barycenter of each face. We subsequently remesh  $A$  to gain a mesh  $\tilde{A}$  whose edges follow the frame field. This is a nontrivial task which we however do not consider a contribution of the present chapter. We employed the method of [15], which is a version of mixed-integer quadrangulation [16]. Once  $\tilde{A}$  is known, we remesh  $B$  by applying the correspondence  $A \longleftrightarrow B$  to vertices of  $\tilde{A}$ , which yields vertices of  $\tilde{B}$  (Fig. 2.7b).

**Optimization of Support Structures.** The preceding paragraphs show how to find corresponding meshes  $\tilde{A} = (\tilde{V}, \tilde{E}, \tilde{F})$  and  $\tilde{B}$  which represent an almost-torsal parametrization  $\tilde{\mathcal{L}}$  of the congruence  $\mathcal{L}$  (Fig. 2.7b). We optimize  $\tilde{A}, \tilde{B}$  such that corresponding edges become co-planar:

$$\tilde{\mathbf{a}}_i, \tilde{\mathbf{a}}_j, \tilde{\mathbf{b}}_i, \tilde{\mathbf{b}}_j, \text{ co-planar, whenever } \tilde{\mathbf{a}}_i \tilde{\mathbf{a}}_j \in \tilde{E}.$$

We wish to achieve this while retaining proximity of  $\tilde{A}$  to the reference surface  $A$ , and likewise retaining proximity of  $\tilde{\mathcal{L}}$  to the reference congruence  $\mathcal{L}$ . We therefore minimize

$$\begin{aligned} \tilde{f} &= \tilde{w}_{\text{plnr}} \frac{1}{\delta^2 |\tilde{E}|} \sum_{\tilde{\mathbf{a}}_i \tilde{\mathbf{a}}_j \in \tilde{E}} \text{dist}(\tilde{\mathbf{a}}_i \vee \tilde{\mathbf{b}}_j, \tilde{\mathbf{a}}_j \vee \tilde{\mathbf{b}}_i)^2 \\ &+ \tilde{w}_{\text{prox}} \frac{1}{|\tilde{V}|} \sum_{\tilde{\mathbf{a}}_i \in \tilde{V}} \left( \|\tilde{\mathbf{e}}_i - \tilde{\mathbf{e}}_{i,0}\|^2 + \left( \frac{\text{dist}(\tilde{\mathbf{a}}_i, A)}{\delta} \right)^2 \right), \\ &+ \tilde{w}_{\text{fair}} \frac{1}{\delta^2 |\tilde{V}|} \sum_{\tilde{\mathbf{a}}_i \in \tilde{V}} \|\Delta \tilde{\mathbf{a}}_i\|^2. \end{aligned} \quad (2.8)$$

Here the first summand (with weight  $\tilde{w}_{\text{plnr}} \geq 0$ , normalized by an average edge length

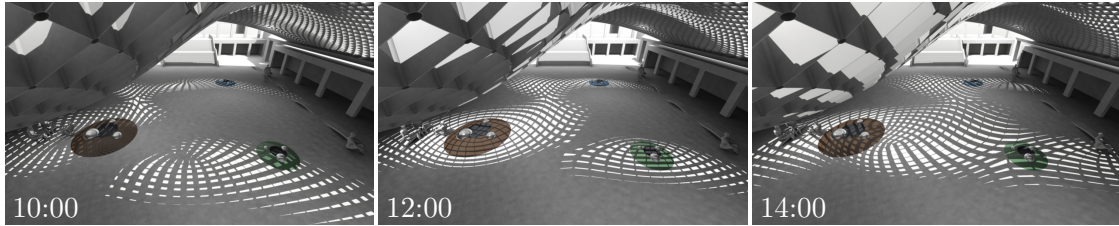


Figure 2.8: Selective Shading: Moving patterns generated by shading system optimized for blocking light at 12:00 except at designated areas.

$\tilde{\delta}$ ) penalizes deviation of quadrilaterals  $\tilde{\mathbf{a}}_i\tilde{\mathbf{a}}_j\tilde{\mathbf{b}}_i\tilde{\mathbf{b}}_j$  from planarity by computing the sum of squares of distances of their diagonals.

The second summand (weighted with  $\tilde{w}_{\text{prox}}$ ) penalizes deviation of the vectors  $\tilde{\mathbf{e}}_i$  (indicating lines of  $\tilde{\mathcal{L}}$ ) from their initial values  $\tilde{\mathbf{e}}_{i,0}$ ; and deviation of vertices  $\tilde{\mathbf{a}}_j$  from the reference surface  $A$ . Here the symbol  $\text{dist}(\mathbf{a}_i, A)$  does not really mean the distance from  $A$  (which is hardly efficiently computable), but an approximation of that distance by  $\text{dist}(\mathbf{a}_i, T_i)^2$ , where  $T_i$  is an estimated tangent plane of  $A$  in the point which arises by closest-point projection onto  $A$  of the position of  $\mathbf{a}_i$  in the previous round of iteration. See Fig. 2.7c, and see Figure ?? for details on the choice of weights.

### 2.4.3 Results

We apply §2.4.1, §2.4.2 to shading systems with both planar and developable elements, and also to indirect lighting.

**Shading Systems for Facades** (*Figure 2.6*). In each of these examples a congruence  $\mathcal{L}$  is optimized so that a torsion-free support structure extracted from it blocks the rays of the sun during the hottest parts of the day. The astronomical information necessary to perform such computations is easily obtainable, since the path of the sun throughout the year is known. For optimization we simply employ directions of light which correspond to the location of the sun during “hot” times like early afternoon in summer. If the depth of shading fins is made minimal, then obviously at other times the sun is not completely blocked. Note that these shading systems are “freeform”

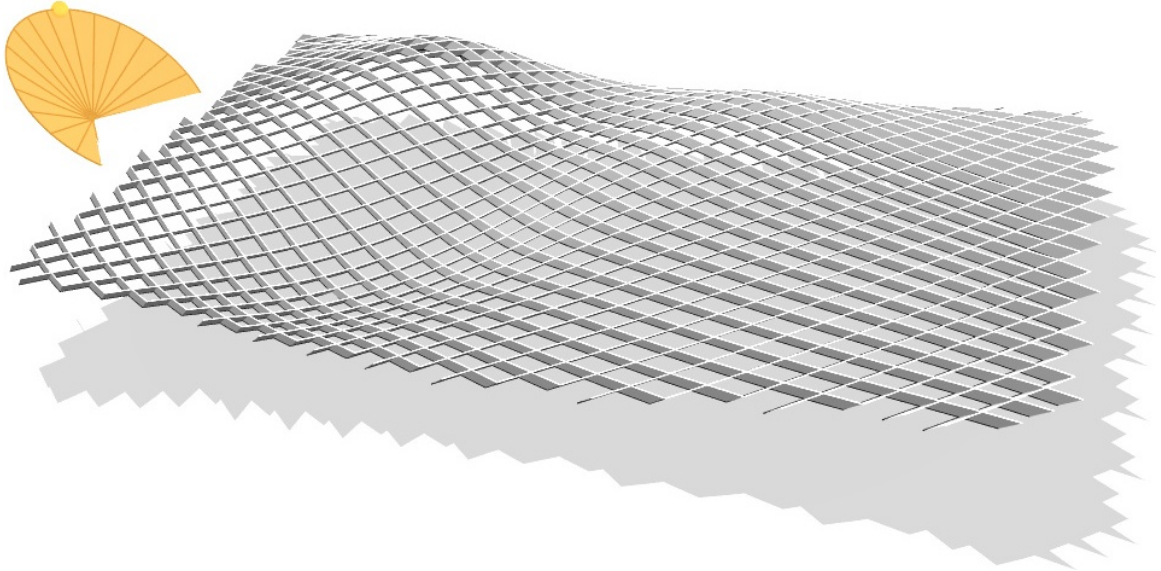


Figure 2.9: Creating full shade by thin developable strips, created by the application of subdivision+optimization to a shading system with planar faces.

even if the underlying reference surface is not, such as in Figure 2.6d.

For Figure 2.6 in general, a design surface (referred to as “mesh  $A$ ” in our description of the optimization procedure) is equipped with a line congruence  $\mathcal{L}$ , which is initialized from surface normals of  $A$  and is subsequently optimized using a target functional composed of  $f_{\text{norm}}$ ,  $f_{\text{fair}}$ , and a linear combination of terms  $f_{\text{perp},i}$  (among other terms). The latter make lines of  $\mathcal{L}$  orthogonal to the vector  $\mathbf{d}_i = \mathbf{d} = \text{const}$  which indicates the direction of light. Having computed  $\mathcal{L}$ , we perform quad remeshing guided by  $\mathcal{L}$ ’s torsal frame field, and subsequently optimize a torsion-free support structure.

For Figures 2.6a, 2.6b the support structure is to be aligned with the boundary and a user’s design strokes, so optimization uses terms  $f_{\text{dir}}(\Delta)$  to achieve prescribed torsal directions for faces contained in a certain subset  $F' \subseteq F$  ( $F'$  is marked in red in small inset figures). In a similar manner the shading system of 2.6d has been optimized. As an alternative to  $f_{\text{perp},i}$ , here sun blocking is achieved using a linear combination of terms  $f_{\text{plane}}(\Delta)$  which position torsal planes of the congruence directly orthogonal to incoming light.

Finally Figure 2.6c exhibits a shading system with the property that certain objects are visible through the shading system in designated areas (blue rectangles in inset figure). Optimization therefore has to make sure that for vertices in a subset  $V' \subseteq V$  the lines of  $\mathcal{L}$  pass through prescribed target points  $O_i$ . This constraint is incorporated into our optimization by augmenting the target functional with linear combinations of  $f_{\text{par},i}$ , for vertices in  $V'$ . Such constraints could be used e.g. for ensuring that people in offices see a portion of the sky. —newFor optimizing the congruences  $\mathcal{L}$  corresponding to Figure 2.6, we use the target functional

$$\begin{aligned} f &= w_{\text{norm}} f_{\text{norm}} + w_{\text{fair}} f_{\text{fair}} + w_{\text{fair/t}} f_{\text{fair/t}} \\ &+ w_{\text{perp}} \frac{1}{|V|} \sum_{\mathbf{a}_i} f_{\text{perp},i} + w_{\text{dir}} \frac{1}{|F'|} \sum_{\Delta \in F'} f_{\text{dir}}(\Delta) \\ &+ w_{\text{par}} \frac{1}{|V'|} \sum_{\mathbf{a}_i \in V'} f_{\text{par},i} + w_{\text{plane}} \frac{1}{|F|} \sum_{\Delta \in F} f_{\text{plnr}}(\Delta). \end{aligned} \quad (2.9)$$

**Selective Blocking of Light** (*Figure 2.8*). This architectural design is to give shade except for a designated area where shading fins are to be parallel to incoming rays. To create this example we proceed similar to Figure 2.6c: The base mesh represents the design surface, its normals initialize  $\mathcal{L}$ . A subset  $V' \subset V$  of vertices specifies the area where light should come through. The optimization uses the target functional (2.9), with the ‘parallel’ and ‘perpendicular’ terms given as

$$\frac{w_{\text{par}}}{|V'|} \sum_{\mathbf{a}_i \in V'} f_{\text{par},i} + \frac{w_{\text{perp}}}{|V \setminus V'|} \sum_{\mathbf{a}_i \in V \setminus V'} f_{\text{perp},i}.$$

( $f_{\text{par},i}$ ,  $f_{\text{perp},i}$  involve the direction  $\mathbf{d}_i = \mathbf{d} = \text{const}$  of light).

**Shading by Single-Curved Elements** (*Figure 2.9*). A sequence of planar quadrilaterals is a discrete developable surface (see e.g. the red and yellow developables of Figure 2.4). This interpretation motivates us to apply a refinement procedure according to [9] to a torsion-free support structure in order to convert it into a system of *smooth developables*: we iteratively apply splitting, smoothing, and optimization

towards planar faces. Applications are structures built from plywood or sheet metal, whose manufacturing depends on the developability property.

**Indirect Lighting by Reflection** (*Figures 1.1, 2.10, 2.11*). We extend our methods to indirect lighting by reflection. To guide a ray of light towards a prescribed direction, a bisector plane of the original ray and the reflected ray has to be used as a mirror surface. We therefore optimize a congruence  $\mathcal{L}$  such that precomputed mirror planes become torsal planes. Figure 2.10 actually exhibits a 2nd torsion-free supporting structure, which does have the function indicated by its name, namely a steel substructure aligned with the shading system (Fig. 2.10(d3)). It is based on a congruence  $\mathcal{L}'$  which is optimized simultaneously with  $\mathcal{L}$ . Alignment of  $\mathcal{L}, \mathcal{L}'$  means that torsal directions of  $\mathcal{L}, \mathcal{L}'$  coincide which is achieved by augmenting the target function (2.9) by

$$w_{\text{extra}} \cdot \frac{1}{|F|} \sum_{\Delta \in F} \left( \frac{1}{\delta^2} \left\| \begin{pmatrix} \gamma_0 \\ 2\gamma_1 \\ \gamma_2 \end{pmatrix}_{\mathcal{L}} \times \begin{pmatrix} \gamma_0 \\ 2\gamma_1 \\ \gamma_2 \end{pmatrix}_{\mathcal{L}'} \right\| \right)^2$$

( $\gamma_i$  are the coefficients of (2.1), evaluated in face barycenters).

## 2.5 Discussion

**Implementation Details.** Details on optimization for the examples contained in this chapter are given in Figure ???. In particular we give the quality of planarity for each occurring torsion-free support structure. We found that planarity is mostly sufficient already after the extraction procedure of §2.4.2 so no further optimization is needed. Timings are for a 2.4GHz dual core desktop with 6GB RAM.

**Limitations.** We generally found in our examples that we could successfully optimize congruences as desired. However it is not possible to fulfill our kind of geometric side-conditions for *normal* congruences. As a consequence, the contribution  $f_{\text{norm}}$  to the

target functional works as a regularizer and more importantly, it makes congruences hyperbolic and therefore usable for support structures. A general limitation of static systems which block or guide light from moving sources is, of course, that they are optimal only for the few positions of the light source they have been optimized for; see Figures 2.6d and 2.8.

**Robustness.** We take as evidence for robustness of our nonlinear optimization procedures that we could initialize congruences from lines orthogonal to the reference surface, even if the result of optimization is far from orthogonal. Experiments show that adding noise (uniformly distributed, up to maximum  $\approx 50^\circ$ ) does not visibly influence the result.

**Alternative Routes.** We employ a two-step procedure: optimization of a congruence and subsequent extraction of a torsion-free support structure (which determines the combinatorics of the shading system). The separation into these two steps is essential: we found that the simpler method of directly optimizing vertex positions of a no-optimal shading system does not work. An alternative approach is to determine the orientation of torsal planes from the desired light pattern. For smooth congruences 1 family of torsal planes determines the congruence including the 2nd family of torsal planes (by differentiating twice). This method works in principle, but we found it not very robust.

## 2.6 Conclusion

This chapter demonstrates geometric basics and applications of discrete congruences, with a focus on shading and lighting systems. Our procedures can be applied to any kind of geometry, from flat to double-curved. We thus combine an area deeply rooted in graphics (i.e., shading and lighting) with geometric computing and optimization in architectural design. Directions for future research are many: discrete curvatures

defined in terms of normal congruences are a topic of discrete differential geometry. Other topics have to do with manufacturing and assembly, e.g. beams of constant height, more general shapes as shading elements etc.



Figure 2.10: Shading and lighting systems for a subway entrance in London at noon, June 21. Rows a–d represent top view, front view, cross-section, and interior view, respectively. Columns 1–4 correspond to different shading systems extracted from a congruence  $\mathcal{L}$ : In (a1)–(c1)  $\mathcal{L}$  consists of the normals of the mesh  $A$  which represents the roof, so the support structure follows the principal curvature lines of  $A$ . There is no effective shading. In (a2)–(c2)  $\mathcal{L}$  is optimized for blocking sunlight. In (a3)–(e3)  $\mathcal{L}$  is optimized such that shading fins reflect sunlight towards the interior at an angle of 45 degrees, with (e3) illustrating reflected light rays. In (a4)–(e4) the front part of the roof is optimized for shading, the rest for reflection.



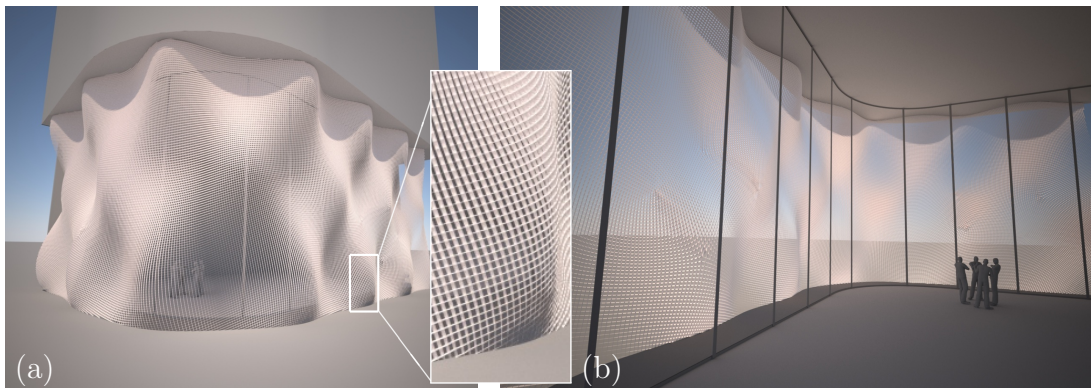


Figure 2.11: (a) A surface equipped with a torsion-free support structure effecting indirect lighting by reflecting sunlight onto the ceiling. (b) If made from a non-reflecting material, shading and diffuse reflecting lighting is achieved. The screen is almost transparent when viewed from the inside (“veil of light”).

## Chapter 3

# Freeform Honeycomb Structures

### 3.1 Introduction



Figure 3.1: Left: Natural honeycomb. Right: Design by E. van Egeraat based on a hexagonal torsion-free support structure.

Nature's design strategies and solutions are a rich source of inspiration for various branches of science and technology (see e.g. the biomimicry web pages [www.ask-nature.org](http://www.ask-nature.org) and [biomimicry.net](http://biomimicry.net)). Architecture and structural engineering are certainly among those areas which learn from nature, and this should be especially true for research on the realization of complex architectural structures. The present chapter presents a contribution in this direction. It is inspired by *honeycombs* which possess fascinating freeform shapes (see Figure 3.1) and are composed of hexagonal cells whose

faces meet at angles close to 120 degrees. Structures containing honeycomb geometry have been used in engineering for a long time, as a means to minimize material without losing structural strength [17]. This chapter assumes a different viewpoint and sees them in the context of architectural geometry, as *torsion-free support structures* with congruent regular nodes and hexagonal cells. The most important property of such honeycomb structures is that all nodes are congruent within a reasonable tolerance, which should facilitate the fabrication of these structures. We will show how to compute and design honeycomb structures, discuss applications and their limitations, and we also discuss the new topic of *polyhedral patterns*.

**Related Work.** Given a base mesh  $M$  of any connectivity, computing a torsion-free structure based on  $M$  requires us to assign a plane to each edge so that the planes around each vertex (*node*) intersect in a straight line, which is called the *node axis*. In the actual realization of this structure, beams are positioned symmetric to the edge planes. For triangle meshes, torsion-free support structures have very few degrees of freedom and are not interesting for applications. This is because all node axes need to pass through a fixed point, or are parallel [18]. For meshes with planar faces, torsion-free support structures have been well studied in recent years: They are accessible via the concept of *parallel meshes*. For quad meshes, support structures are related to *discrete line congruences* [19]. Neither applies to honeycomb structures, which are based on hex meshes with non-planar faces.

Special hex-dominant torsion-free structures can be derived from triangle meshes with the circle-packing property [20] which are equipped with a packing of spheres centered in the vertices. The sphere's tangent planes in the points of contact form a torsion-free hex structure, and if all spheres are of the same size it will be a honeycomb with 120 degree intersection angles. Such CP meshes therefore are useful to initialize some optimization tasks discussed in this chapter. A more general construction of support structures has been briefly addressed in connection with cell packing

structures [21], but without aiming at congruent nodes.

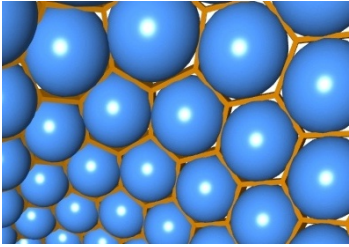


Figure 3.2: Hexagonal support structure derived from a circle-packing mesh, each cell containing a sphere touching the walls [20].

*Repetition of elements* has been a successful ingredient in reducing the fabrication cost of freeform architecture, see e.g. [22], and thus this topic already received some attention. Singh and Schaefer [23] optimize triangle meshes so that there is only a relatively small set of different faces up to some chosen tolerance. Similarly, Fu et al. [24] reduce the number of essentially different faces in non-polyhedral quad meshes. For arbitrary freeform shapes, the goal of congruent faces appears to conflict mesh fairness. We expect a similar effect for the goal of congruent nodes in meshes with straight edges. However, giving up the straightness of edges and using circular arcs instead, one can achieve congruent and even regular nodes for triangular, quad and hex combinatorics [25].

Last, but not least we point to [26], where the topics of repetition and general polyhedral patterns are combined by the analysis of the realization of surfaces with a bounded number of “folding elements”.

## Contributions

- (A). We analyze the geometry of honeycomb structures and discuss their flexibility to approximate freeform shapes. It turns out that a honeycomb can be orthogonal to a given surface only if that surface is developable. The Gaussian image of a honeycomb is an unusual kind of spherical 2D tiling which is concentrated on curves.
- (B). We present computation and interactive design of honeycombs, addressing spe-

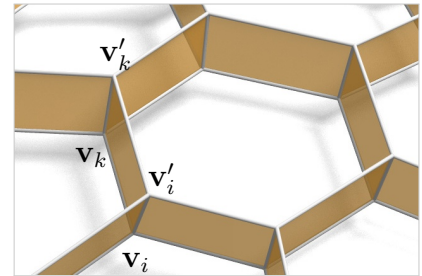
cial cases and particular applications relevant to architectural design.

- (c). Honeycombs can be converted to polyhedral patterns on surfaces, e.g. quad patterns consisting of planar faces. Such patterns, which are *not* a discrete version of a smooth curve network, constitute a remarkable and novel topic.

## 3.2 Geometry and flexibility of honeycomb structures

This section discusses geometric properties of honeycombs. Our conclusions in particular yield insights in the degrees of freedom which are available when approximating a design surface by a honeycomb structure. It turns out that the condition of *congruent nodes* is rather strong: It implies that the honeycomb can always follow a given surface, but in general it is not possible that the node axes remain orthogonal to it. We show that this can happen only for developable surfaces.

**Definitions.** We summarize again the definition of a honeycomb structure: It requires two combinatorially equivalent meshes  $M = (V, E, F)$  and  $M' = (V', E', F')$  such that each pair  $\mathbf{v}_i \mathbf{v}_k, \mathbf{v}'_i \mathbf{v}'_k$  of corresponding edges defines a planar quadrilateral



*wall* which serves to border the open cells associated with the faces of the mesh. Further we require that for each vertex the incident walls form angles of 120 degrees (implying that the valence of each vertex is  $\leq 3$  and faces will be mostly hexagonal). The intersection of walls at a vertex  $\mathbf{v}_i$  is called its *node axis* (it is the straight line  $\mathbf{v}_i \mathbf{v}'_i$ ).

The most relevant geometric information contained in a honeycomb structure is not the meshes  $M, M'$  but the edge planes which carry the walls (connecting corre-

sponding edges), and the node axes which connect corresponding vertices. If these data are given we may freely choose vertices of  $M, M'$  in the node axes, thereby defining a honeycomb structure (this e.g. implies that any surface can trivially be approximated by a honeycomb structure, by moving the vertices of an existing honeycomb close to that surface).

We will see that for our computations it is mostly sufficient to encode the relevant geometric data of honeycombs by the vertices of the mesh  $M$ , and the normal vectors of walls in the honeycomb.

**The spherical image of a honeycomb.** In order to understand the global geometry of honeycombs we study the unit vectors  $\mathbf{v}_i^\sigma$  which indicate the directions of the node axis  $\mathbf{v}_i \mathbf{v}_i'$ . In our applications the base mesh is always following a smooth surface, and we assume that the vectors  $\mathbf{v}_i^\sigma$  consistently point to one side of that surface. For any face  $f = (\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \dots)$  of the base mesh, the *spherical image* of that face, resp. cell, is the spherical polygon  $f^\sigma = (\mathbf{v}_{i_1}^\sigma, \mathbf{v}_{i_2}^\sigma, \dots)$ , see Figure 3.3. The spherical image  $M^\sigma$  of the honeycomb consists of the spherical images of the individual cells.

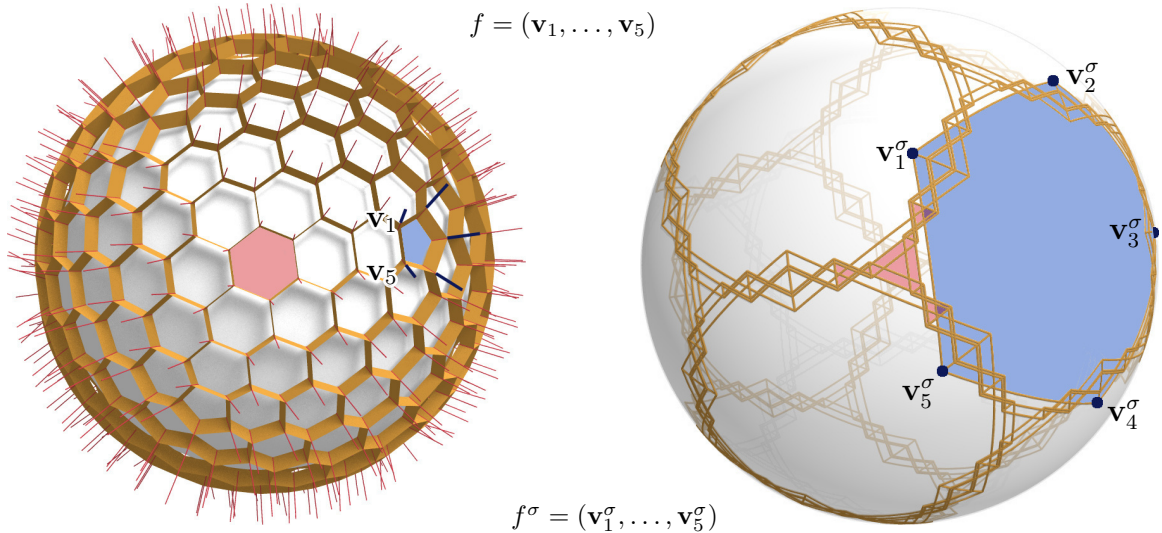


Figure 3.3: Honeycomb and its spherical image. *Left:* Honeycomb with node axes. A 6-gon and a 5-gon are highlighted in red and blue, resp. *Right:* Spherical image  $M^\sigma$  of node axes, with analogous highlighting. Observe the decomposition of  $M^\sigma$  into curve-like structures.

*Remark:* There is a linear space of honeycomb structures which belong to a given spherical image. The dimension of that space is  $|E| - |V| \approx \frac{1}{2}|V|$ , because there is one degree of freedom per edge plane, and one condition per vertex.

**The spherical zero-area property of honeycombs.** Since the walls of a honeycomb cell intersect at 120 degrees, the spherical image  $f^\sigma$  of this cell is a polygon whose edges are great circles intersecting at  $60^\circ$  or  $120^\circ$ . Figure 3.3 exhibits two kinds of cells:  $f^\sigma$  may be large (like the highlighted 5-gon), or it might be small, meaning that the node axes of the corresponding cell point in roughly the same direction.

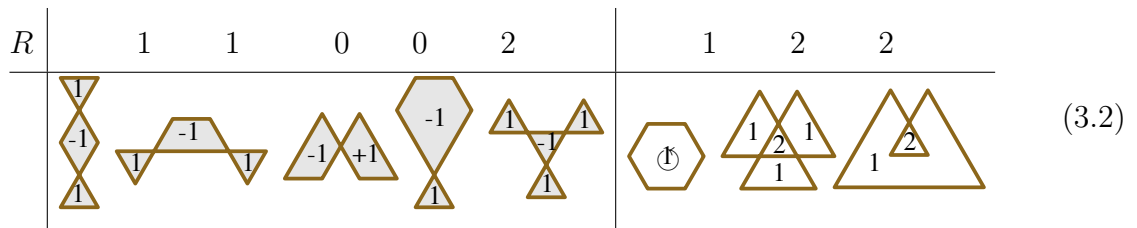
Recall the Gauss-Bonnet formula which relates the total rotation number  $R$  of a geodesic polygon in a surface with the Gauss curvature  $K$  and the angles  $\alpha_i$  which indicate the turning of the edge in the individual vertices. We have

$$\sum \alpha_i = 2\pi R - \int_x K(x)w(x), \tag{3.1}$$

where  $w(x)$  is the number of times the polygon is winding around the point  $x$ .

*Example:* For the blue polygon  $f^\sigma$  in Figure 3.3 we have  $K = 1$ , all  $\alpha_i$  equal  $60^\circ$ ,  $R = 1$ , and the winding coefficient  $w(x)$  equals 1 inside the polygon, and 0 outside. Thus  $\int K(x)w(x)$  reduces to the area of the polygon, and we get  $5 \times \frac{\pi}{3} = 2\pi - \text{area}(f^\sigma) \implies \text{area}(f^\sigma) = \frac{\pi}{3}$ .

We sketch some *planar* 6-gons whose angles are  $60^\circ$  or  $120^\circ$ , indicating  $R$  and the local winding number  $w(x)$ :



The turning angles  $\alpha_i$  in vertices obey  $\sum \alpha_i = 2\pi R$ , which follows from (3.1) when we let  $K = 0$ . A *spherical* 6-gon  $f^\sigma$ , on the other hand, satisfies Gauss-Bonnet with

$K = 1$ :

$$\sum \alpha_i = 2\pi R - \int_x w(x) = 2\pi R - \text{area}(f^\sigma), \quad (3.3)$$

where  $\text{area}(f^\sigma)$  is the oriented area of  $f^\sigma$ , defined as  $\int w(x)$ , meaning that each point contributes to the area with a certain multiplicity defined by how often the polygon winds around that point. If  $f^\sigma$  has the same combinatorics and angles as a planar hexagon, then  $\sum \alpha_i = 2\pi R$  immediately implies

$$\text{area}(f^\sigma) = 0. \quad (3.4)$$

There may be  $f^\sigma$ 's whose area does not vanish,  $\sum \alpha_i \neq 2\pi R$ , and we cannot find a planar 6-gon with the same angles and rotation number  $R$ . However the smallest nonzero value of  $|\text{area}(f^\sigma)|$  equals  $\pi/3$  because both  $\sum \alpha_i$  and  $2\pi R$  are integer multiples of that number (which, incidentally, is the area of the highlighted 5-gon in Figure 3.3). We have proved:

**PROPOSITION 3.** *The oriented area of a spherical image  $f^\sigma$  of a honeycomb cell either vanishes or has absolute value  $\geq \pi/3$ . In the first case there is a planar hexagon having the same angles  $\alpha_i$  and rotation number  $R$  as  $f^\sigma$  does.*

The practical implication of this statement is that a 6-gon honeycomb cell whose node axes are close together will have a spherical image  $f^\sigma$  of zero area. Figure 3.3 shows an example of this: All hexagons in the spherical image have zero area, and their shapes correspond to the examples in the left hand section of Equ. (3.2).

**Global distribution of node axes.** Numerical evidence (see e.g. Figures 3.3 and 3.4) shows that the spherical image of a honeycomb exhibits curve-like structures where zero-area hexagons cluster, interspersed with individual polygons of nonzero area (non-hexagonal ones mainly, but also hexagons are possible). We argue why *smoothness* of honeycombs is responsible for this behaviour.

This smoothness is not meant in the literal sense, since the shape of spherical



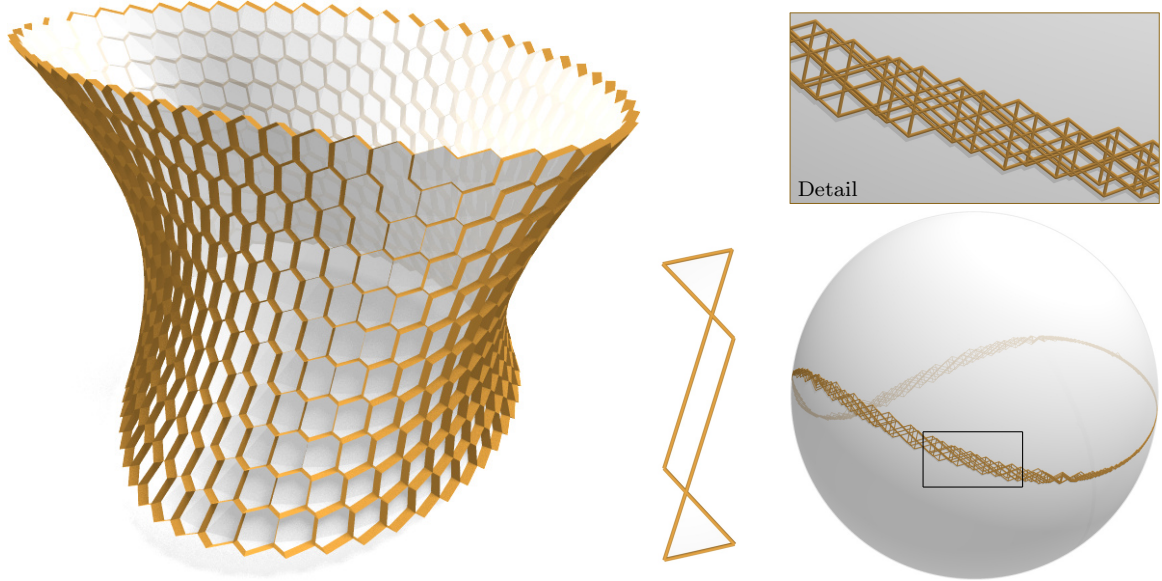
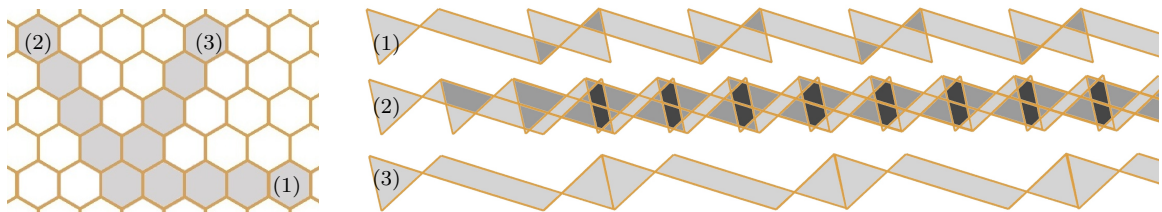





Figure 3.4: This curve-like spherical image (right) of a honeycomb (left) has the aspect of a tiling of the sphere by zero-area hexagons shaped like the one shown above.

image polygons entails high-frequency oscillation of node axes in any case. Instead we require a very modest kind of regularity: In a “regular” area of the honeycomb, the spherical images of two neighbouring cells should have roughly the same size and shape. The tiling of cells of the honeycomb must translate to a tiling of hexagons on the unit sphere. We therefore have to ask which of the hexagons shown in Equation (3.2) can possibly have a similar-shaped hexagon as a neighbour, along each of its six edges.

For polygons of type  $\bowtie$  (also depicted in Figure 3.4), this is possible. There is a combinatorially regular two-dimensional tiling of such hexagons which is difficult to visualize, but which can be seen in Figure 3.4. Three individual rows of hexagons contained in such a tiling are shown below, at right. For comparison we also show three rows of hexagons in the tiling with a geometrically regular hexagon (at left).



Geometrically, this tiling exists because the sum  $\alpha_1 + \alpha_3 + \alpha_5$  of turning angles is an integer multiple of  $360^\circ$ . For the polygons , such a tiling does not exist, because if they are made to have zero area, there is at least one pair of opposite edges of very different lengths. For polygons of type , opposite edges do have the same length, but an attempt at tiling will still fail, because the above-mentioned angle condition is not satisfied.

The image of the three rows also shows that a tiling of hexagons of type  is covering only a curve-like strip (see also Fig 3.4). For the planar version of this hexagon this is because the 6 parallel translations which map an edge to its opposite edge all go in the same direction; for the spherical version of this hexagon this property holds only approximately. We summarize:

PROPERTY 4. *The spherical image of a regular honeycomb is curve-like.*

The practical implication of this statement is that node axes of a honeycomb will not behave like the normals of a surface, even if high-frequency oscillations are discarded. Only for *developable* surfaces, the spherical image of normals is curve-like (see Figure 3.5). This implies:

PROPERTY 5. *Node axes of a regular honeycomb approximate the normals of a surface only if that surface is developable.*

*Remark:* We also ask the converse question: Can we find a “zero-area” tiling along a given spherical curve  $C$ , which is the spherical image of a honeycomb structure? The answer is yes by the following heuristic argument: we start with a zigzag polyline following  $C$  and extend it by adding zero-area hexagons layer by layer. For each new row there are as many degrees of freedom (lateral displacement of vertices) as there are conditions (zero area for each face).

If a honeycomb is guided by a surface  $S$  but the node axes are *not* required to be close to the normals of  $S$ , then there are many design options. When honeycombs are

smooth (in an appropriate meaning of the word) then node axes will automatically be close to the normals of some developable, but that developable is present only implicitly and does not interfere with computations.

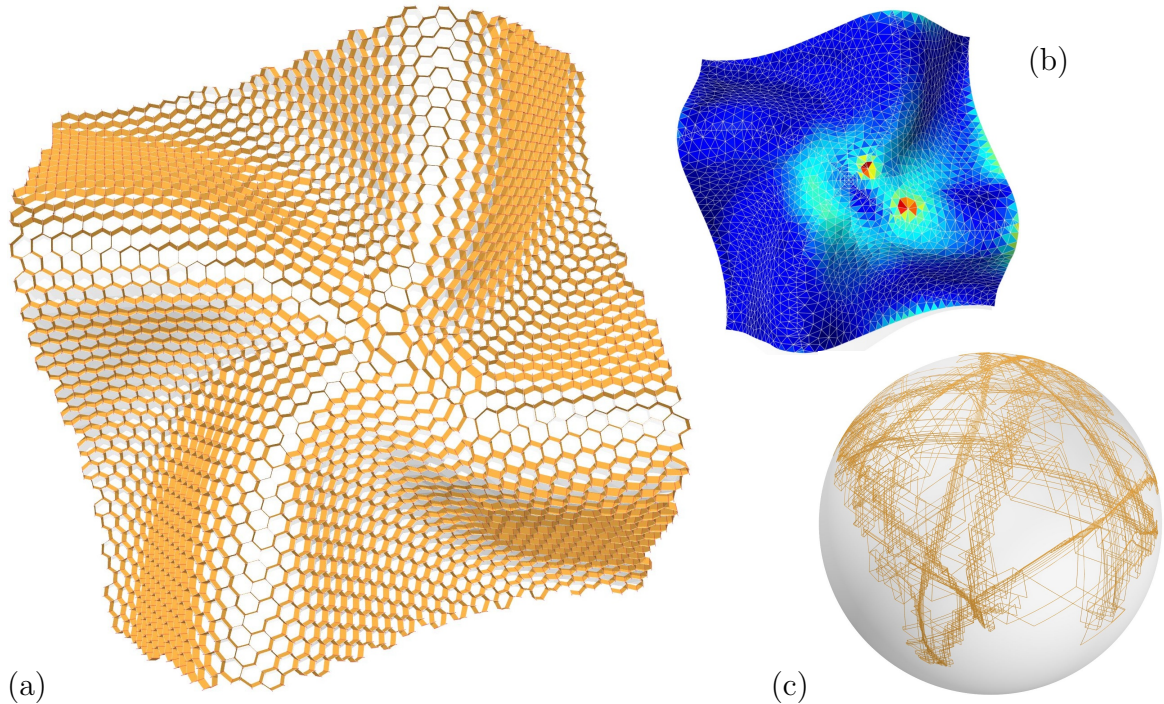


Figure 3.5: (a) Honeycomb following a near-developable surface  $S$  (which is the only kind of surface where node axes can follow the surface normals). (b) Color coded quality of a near-isometric parametrization of  $S$ , which has been used for initializing the honeycomb, by mapping a regular hexagonal tiling onto  $S$  (distortion peak .30 and average .05). (c) Spherical image of honeycomb. This result experimentally confirms that our method of initialization provides almost-isometric mappings; for details see text.

### 3.3 Computational approach

Here we describe our computational setup for honeycomb structures. We discuss the two processes of (i) finding an initial hex mesh equipped with additional data which do not yet define a torsion-free support structure, and (ii) optimizing these data so they will define a honeycomb structure.

*Initialization of the base mesh.* For initialization, we implemented the method of [27]

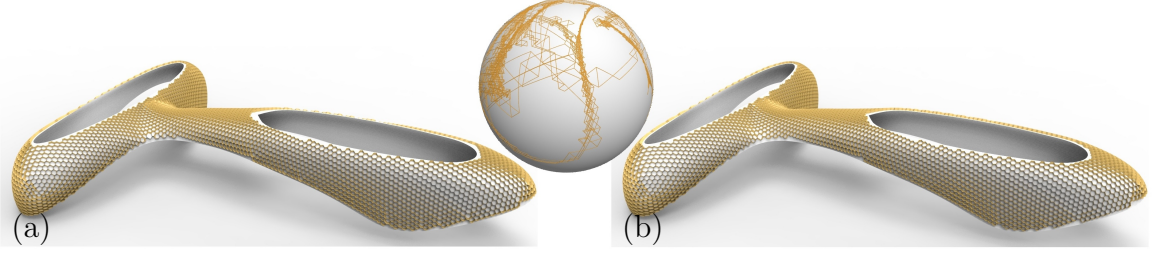
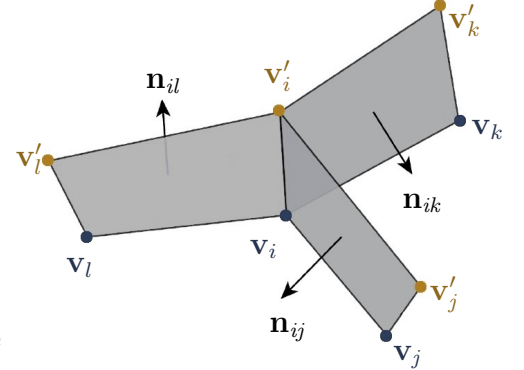


Figure 3.6: Computing a honeycomb on a complex architectural freeform skin (Yas Marina hotel, Abu Dhabi). (a) Given the base mesh, we initialize a honeycomb structure (not yet consistent) orthogonal to the reference surface and apply guided projection onto the constraint manifold. (b) This result is not satisfactorily regular, so the projection procedure has to be guided by a fairness energy. We added equality of normal vectors of opposite walls in each hexagonal cell as soft constraints (with the same weight as the regularizing term) to our algorithm. Note that the spherical image of this honeycomb exhibits Property 2.

to find a hexagonal mesh  $M = (V, E, F)$  on a given surface  $S$ , whose edges follow a given 6-RoSy field of directions. To find that field, we use a direct extension of the method of [28] which can easily accommodate alignment with features or a user's design strokes.

**Choice of variables.** The variables used in optimization are the vertices together with unit vectors  $\mathbf{n}_{ij}$  for each edge  $\mathbf{v}_i\mathbf{v}_j$ . In the following text we assume that  $\mathbf{n}_{ji} = -\mathbf{n}_{ij}$  but of course only one of  $\mathbf{n}_{ij}$ ,  $\mathbf{n}_{ji}$  is actually used in the implementation. These vectors are initialized by letting  $\mathbf{n}_{ij} = R_{-\pi/2}(\frac{\mathbf{v}_i - \mathbf{v}_j}{\|\mathbf{v}_i - \mathbf{v}_j\|})$ , with  $R_{-\pi/2}$  as a clockwise rotation by 90 degrees about a surface normal in the midpoint of the edge  $\mathbf{v}_i\mathbf{v}_j$ . The vectors  $\mathbf{n}_{ij}$  are intended to serve as normal vectors of walls in the honeycomb, meaning that for each vertex  $\mathbf{v}_i$  there should exist a node axis  $\mathbf{v}_i\mathbf{v}'_i$  which is orthogonal to  $\mathbf{n}_{ij}$  whenever  $\mathbf{v}_i\mathbf{v}_j$  is an edge.



*Reconstructing the honeycomb structure from the chosen variables.* This is done in the following way: We must find the vertices  $\mathbf{v}'_i$  of the mesh  $M'$  paired with  $M$ , by letting  $\mathbf{v}'_i = \mathbf{v}_i + \lambda_i\mathbf{v}_i^\sigma$ , where  $\lambda_i$  is the prescribed thickness of the honeycomb and  $\mathbf{v}_i^\sigma = \frac{1}{\|\mathbf{v}'_i - \mathbf{v}_i\|}(\mathbf{v}'_i - \mathbf{v}_i)$  is a unit vector indicating the node axis through  $\mathbf{v}_i$  which

must be orthogonal to all  $\mathbf{n}_{ij}$ 's. It is found by principal component analysis, as an eigenvector of the 3 by 3 matrix  $\sum_j \mathbf{n}_{ij} \mathbf{n}_{ij}^T$ .

*Constraints.* We impose the following constraints on our variables. Firstly, every normal vector is normalized, so  $\mathbf{n}_{ij}^T \mathbf{n}_{ij} = 1$ . Secondly we have the consistency condition

$$\mathbf{n}_{ij}^T (\mathbf{v}_i - \mathbf{v}_j) = 0 \quad \text{whenever} \quad \mathbf{v}_i \mathbf{v}_j \in E.$$

The intersection angle of walls associated with edges  $\mathbf{v}_i \mathbf{v}_j$  and  $\mathbf{v}_i \mathbf{v}_k$  could be expressed by  $\mathbf{n}_{ij}^T \mathbf{n}_{ik} = \cos 120^\circ = -1/2$ . For a valence 3 vertex  $\mathbf{v}_i$  with neighbours  $\mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_l$  however, the simple condition  $\mathbf{n}_{ij} + \mathbf{n}_{ik} + \mathbf{n}_{il} = \mathbf{o}$  states that these three normal vectors form an equilateral triangle. This expresses both the angle condition and the existence of a common node axis. Further constraints are vertices confined to the design surface  $S$  or to a boundary curve. Since the geometry of the honeycomb permits moving vertices along the node axes, proximity to  $S$  can be considered a soft constraint.

*Solution.* Each constraint involves only a few variables and (apart from interpolation constraints) is either linear or quadratic. We therefore use the method of [29] to solve the constraint equations iteratively. In every round of iteration a Newton linearization turns the constraints into a linear system of the form  $HX = R$ , with  $X$  as the increment in the vector of variables. To allow for redundant constraints and the general under-determinedness of the system, we do not solve it directly, but we regularize and instead solve  $\|HX - R\|^2 + \varepsilon \|X\|^2 \rightarrow \min$ , with  $\varepsilon \ll 1$ . Note that our solution procedure is essentially the same as minimizing a least-squares sum of constraints with a Gauss-Newton method.

Interpolation constraints have the form " $\mathbf{v} \in \Phi$ " where  $\Phi$  is a curve (or surface). They are linearized in each round of iteration by replacing  $\Phi$  with the tangent (or tangent plane) in the point  $\mathbf{v}^* \in \Phi$  which is closest to  $\mathbf{v}$ .

*Fairness* is not used in the basic version of our algorithm. It can be incorporated either as a set of soft constraints (with small weights) or by adding a fairness energy to the regularizing term, which amounts to the same thing (see Fig. 3.6).

**Relevance of geometric properties.** Section 4.3 discussed in detail geometric properties of honeycomb structures. Their most important implications on our algorithm (this section) and on applications (next section) are the following:

- (A). We cannot expect that the walls of honeycomb cells can be made orthogonal to a reference surface – this is possible only if that surface is developable (cf. Property 2 and Property 3).
- (B). We cannot expect the node axes (or the cell’s normal vectors) to be smooth in the usual discrete sense of finite differences, since the zero area property of Proposition 1 implies high frequency oscillations.

In particular we conclude that orthogonality of a honeycomb to a surface cannot be a hard constraint, and that fairness energies must be composed appropriately.

The next section discusses applications and extensions of the concept of honeycomb, and also mentions where it is necessary to modify the initialization or to add additional variables and constraints to optimization.

## 3.4 Results

The previous sections already contained examples of honeycombs, see Figures 3.3, 3.4, 3.5 and 3.6. Property 2 is validated by the spherical images of honeycombs shown there. Further, Figure 3.7 gives an impression on how well the constraint equations are satisfied before and after optimization.

**Honeycombs following developable surfaces.** Figure 3.5 shows a honeycomb whose guiding surface  $S$  is nearly developable and where initialization according to

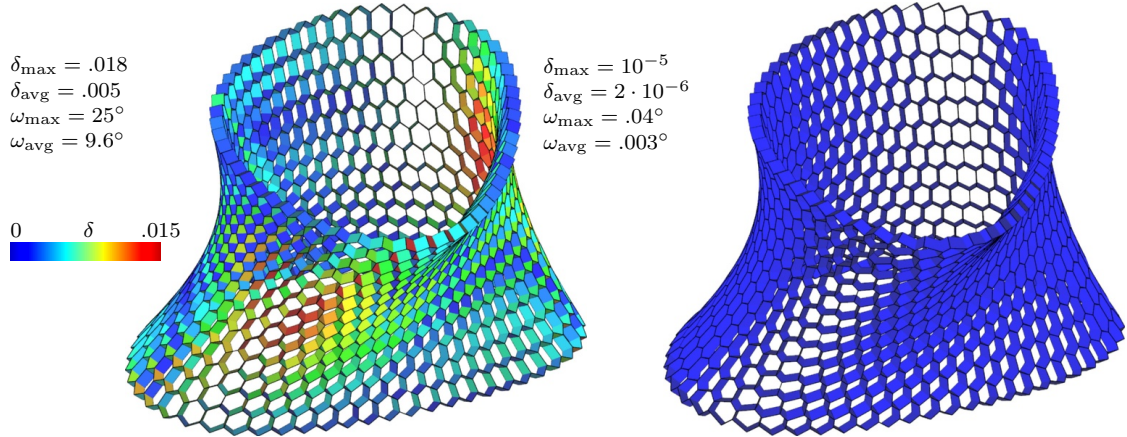


Figure 3.7: We illustrate to which extent the constraints are fulfilled, for the honeycomb of Figure 3.4, after initialization and before optimization (left) and after optimization (right). The planarity of wall quadrilaterals is indicated by color coding the planarity measure  $\delta$ , which is defined as the distance of diagonals of a quad divided by average edge length. “ $\omega$ ” is the deviation from the desired intersection angle of  $120^\circ$ .

§3.3 yields an almost-isometric parametrization of  $S$ . The honeycomb is initialized by mapping a regular hexagonal grid onto  $S$ . For the sake of experiment, we tried to make the parametrization more isometric, following [?] to iteratively modify the 6-RoSy field our method is based on. We found that this improvement has almost no visible effect and that the original method yields a parametrization which is isometric enough anyway.

**User interaction.** Solving the constraint equations necessary for computing a honeycomb is fast enough to enable user interaction. There are three kinds of deformation resp. modeling we emphasize: (i) changing the base mesh and recomputing the hon-

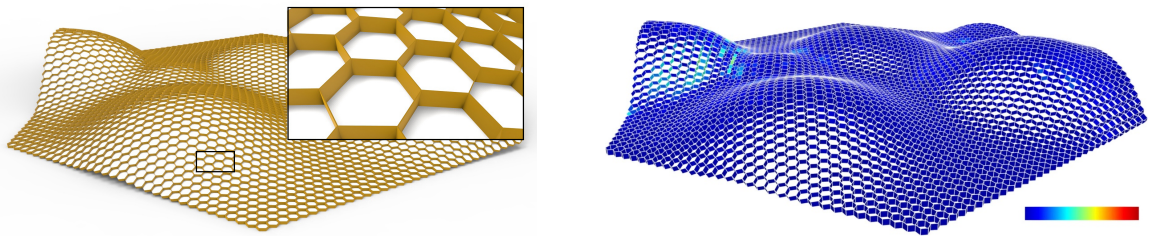


Figure 3.8: Honeycomb defined by parallel meshes  $M$  and  $M'$ . The angle  $\beta$  between corresponding edges  $e, e'$  illustrates deviation from parallelity.

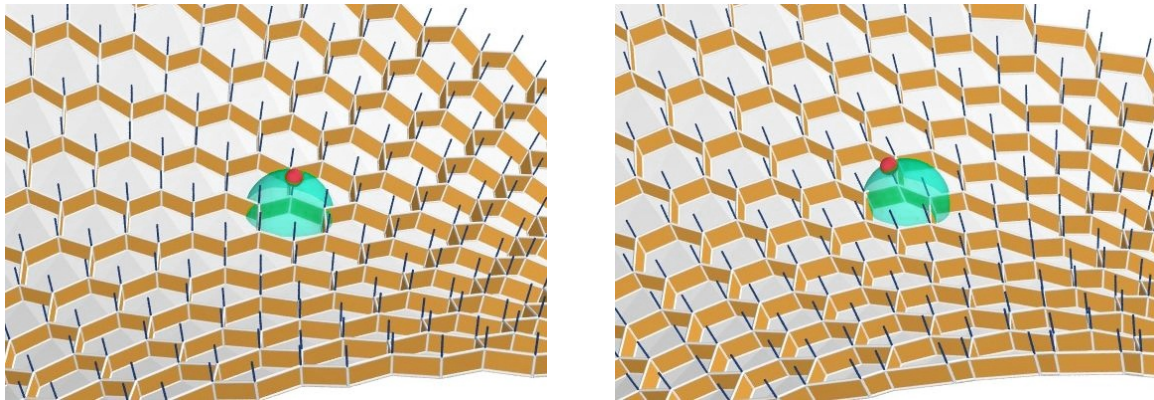


Figure 3.9: An example of user guided design: The user edits the honeycomb of Figure 3.4 by moving a node axis. This change is also applied to neighbouring node axes, multiplied with a dampening function. Re-optimization modifies the honeycomb such that walls intersect near the prescribed node axis positions (see accompanying VIDEO).

eycomb for the modified mesh; (ii) repositioning vertices on the honeycomb’s node axes, so that the wall planes remain unchanged, and (iii) editing the axes with only minimal changes to the base mesh. All three are shown in the accompanying VIDEO. As to computation, editing mode (i) corresponds to setting target values for the vertex coordinates  $\mathbf{v}_i$  and re-running optimization. For (ii), no optimization has to be performed; this deformation is conceptually similar to editing a triangle mesh. For (iii), see Figure 3.9, the user’s wish is translated to new target values  $\mathbf{a}_i$  for the node axis vectors  $\frac{1}{\|\mathbf{v}'_i - \mathbf{v}_i\|}(\mathbf{v}'_i - \mathbf{v}_i)$ . They are incorporated in our optimization by adding constraints

$$\mathbf{n}_{ij}^T \mathbf{a}_i = 0, \text{ whenever } \mathbf{v}_i \mathbf{v}_j \in E.$$

These equations express the wish that the wall planes intersect in the updated node axes. In the optimization these constraints are given a lower weight than those expressing consistency and angles.

**Honeycombs defined by parallel meshes.** We are interested in the question if there are honeycombs defined by *parallel* meshes  $M, M'$ , meaning that corresponding edges of  $M, M'$  are parallel (note that parallelity of this kind is not the same as





Figure 3.10: Zigzag quadrangulation: We show two views of a polyhedral quad mesh equipped with a honeycomb support structure, approximating the design surface of the Cour Visconti roof in the Louvre, Paris (see also Figure 1.2). The right hand image illustrates the “zigzag” mesh polylines (with consecutive vertices  $\mathbf{v}_1, \dots, \mathbf{v}_4$ ) and “straight” mesh polylines (with consecutive vertices  $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$ ). Fairness of these polylines is a topic of optimization.

parallelity of meshes as studied by [18], since  $M, M'$  do not have planar faces). It is not difficult to add this constraint to our optimization. Introducing vertices  $\mathbf{v}'_i$  as additional variables, we add the appropriate consistency condition  $\mathbf{n}_{ij}^T(\mathbf{v}'_i - \mathbf{v}'_j) = 0$  for each edge  $\mathbf{v}_i, \mathbf{v}_j$ , as well as the parallelity constraint  $(\mathbf{v}_i - \mathbf{v}_j) \times (\mathbf{v}'_i - \mathbf{v}'_j) = \mathbf{o}$ . In order to verify that optimization can succeed, we investigate the degrees of freedom available when, given a honeycomb, we try to position the vertices of both  $M, M'$  on their respective node axes such that edges become parallel.

Consider an  $n$ -gon cell, and assume that vertices  $\mathbf{v}_1, \dots, \mathbf{v}_{n-1}$  and  $\mathbf{v}'_1, \dots, \mathbf{v}'_{n-1}$  are chosen already. In order to construct the remaining vertices  $\mathbf{v}_n, \mathbf{v}'_n$  we observe that parallelity of corresponding edges means that the planes  $\alpha, \alpha'$  spanned by  $\mathbf{v}_1, \mathbf{v}_{n-1}, \mathbf{v}_n$  and  $\mathbf{v}'_1, \mathbf{v}'_{n-1}, \mathbf{v}'_n$ , resp., are parallel. Those planes are determined by the data already available:  $\alpha$  passes through  $\mathbf{v}_1, \mathbf{v}_{n-1}$  and is parallel to  $\mathbf{v}'_{n-1} - \mathbf{v}'_1$ , and analogous for  $\alpha'$ . We therefore find  $\mathbf{v}_n$ , resp.  $\mathbf{v}'_n$  by intersecting the given node axis with  $\alpha$ , resp.  $\alpha'$ . This shows

that in a honeycomb of regular combinatorics we can find parallel meshes  $M, M'$ , simply by choosing vertices on a zigzag sequence of walls and working our way outward from there, with 1 d.o.f. per added face.

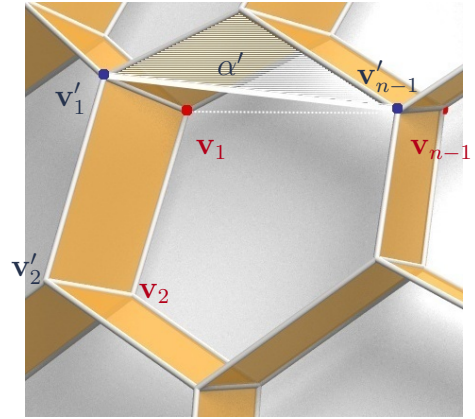


Figure 3.8 shows a result of this kind of optimization. The additional variables  $\mathbf{v}'_i$  have been initialized by moving the vertex  $\mathbf{v}_i$  a constant distance along the normal of the reference surface  $S$ .

**Constant honeycomb depth.** A honeycomb of constant depth is very interesting for architectural geometry, because in this case, we can let beams of constant width follow the edges of  $M, M'$ . It is not difficult to add constant depth as a soft constraint, e.g. by adding quadratic equations  $\|\mathbf{v}_i - \mathbf{v}'_i\|^2 = c$  with a low weight. This turned out to be unnecessary for the example of Figure 3.8.

**Quadrangulation combined with torsion-free support structures.** For applications in freeform architecture the ability to cover a given surface by *planar* elements is particularly relevant. Since a honeycomb structure is usually not bounded by a hex mesh with planar faces, we demonstrate how to cover a honeycomb by a pattern of planar quadrilaterals. The resulting mesh is polyhedral, and is equipped with a torsion-free support structure (which even has the feature of repetition in local node geometry).

Our approach is to split each hexagonal face of the base mesh  $M$  into two quads which are required to be planar; thus capping each cell of the honeycomb with two planar quads. See Figures 1.2, 3.10 for the combinatorics of this splitting and a result.

A major difference to previous approaches to planar-quad meshing is that we do not require smoothness of the edge polygons. In this way we no longer have the strong restrictions posed on quadrangulation which originate in an analogy between meshes and curve networks, cf. [30], e.g. for the surface used in our example, a “smooth”

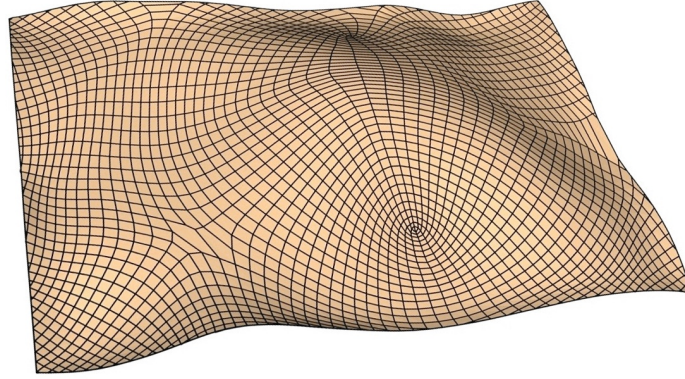


Figure 3.11: It may be difficult to approximate freeform architectural designs (here: Cour Visconti, Louvre) by a quad mesh with planar faces whose edges discretize a smooth network of curves. This is because such meshes have to follow conjugate curve networks, and the number of singularities of these networks is determined by the curvature of the surface.

quadrangulation with planar faces would have to look essentially like the one in Figure 3.11.

In computations we use as additional variables the unit normal vectors of those quads which are required to be planar. The vertices of these quads are already present as variables. Constraints (normalization and consistency) are analogous to the planar “wall” quads.

To capture the visual impression of smoothness (regularity) of both straight and zigzag polylines in the hex mesh  $M$ , we add soft constraints with small weight. Referring to Figure 3.10, a zigzag polyline in a hex mesh is considered regular, if we approximately have  $(\mathbf{v}_1 - \mathbf{v}_2) - (\mathbf{v}_3 - \mathbf{v}_4) = \mathbf{o}$ , for all choices of consecutive vertices  $\mathbf{v}_1, \dots, \mathbf{v}_4$ . For the polylines transverse to the zigzag ones we merely want to punish lateral deviation from the appearance of smoothness, meaning that we want  $\mathbf{n}_{ij}^T(\mathbf{v}_j - \mathbf{v}_k) = 0$  to hold for each choice of 3 consecutive vertices  $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$ . Both kinds of equations are added as soft constraints to our optimization procedure.

*Remark:* It is not difficult to add the condition of a minimum edge length  $l_{\min}$  to our optimization. This can be done by adding quadratic constraints  $\|\mathbf{v}_i - \mathbf{v}_j\|^2 = l_{\min}^2 + (\text{dummy})^2$  which use dummy variables, cf. [29].

**Local edits and subdivision.** The property of two planes forming an angle of  $120^\circ$  is not destroyed by parallel translation. Likewise the intersection line of such planes may move if such a transformation is applied, but it keeps its direction. Our aim is to exploit this fact and modify honeycomb structures by parallel translating the walls. If we wish to keep the property that *three* planes are incident with a node axis, then the translations involved are not independent: If the wall planes associated with edges  $\mathbf{v}_i\mathbf{v}_j$ , resp.  $\mathbf{v}_i\mathbf{v}_k$ , resp.  $\mathbf{v}_i\mathbf{v}_l$  undergo translation by the vector  $\lambda_{ij}\mathbf{n}_{ij}$ , resp.  $\lambda_{ik}\mathbf{n}_{ik}$ , resp.  $\lambda_{il}\mathbf{n}_{il}$ , then a common intersection exists if and only if  $\lambda_{ij} + \lambda_{ik} + \lambda_{il} = 0$  (because the three normal vectors form an equilateral triangle).

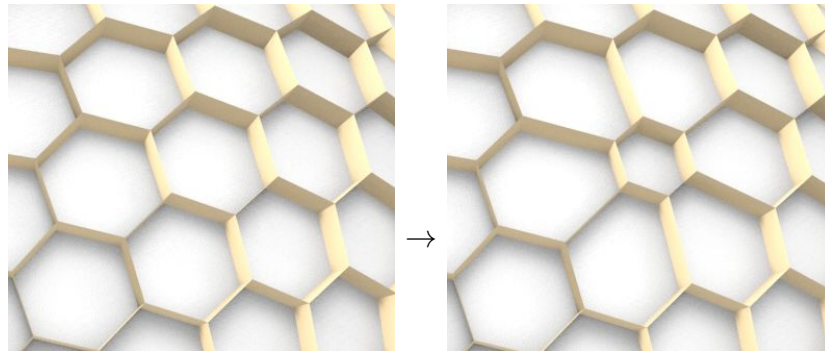


Figure 3.12: Resizing a honeycomb cell by moving walls inwards by the same amount.

One example of such a move is the resizing of a single honeycomb cell (Figure 3.12): here all walls of the cell move by the same amount, while the “radial” walls stay where they are (in the vertex-centered notation used above, we have e.g.  $\lambda_{ij} = 0$ ,  $\lambda_{ik} = -\lambda_{il}$ ).

By resizing *all* cells, we create new hexagonal cells where the original walls had been: this is the subdivision procedure shown by Figure 3.13. Shrinking each cell of the honeycomb in Figure 3.13a to half its size (or another size) produces the subdivided honeycomb of Figure 3.13b (or Figure 3.13c). Its spherical image coincides with the original one, but it has 4 times as many cells.

*Remark:* For a general torsion-free support structure, where planes intersect at

different angles, local edits (and in consequence, subdivision) are not so easy. Each node will have its own condition on the three translations which affect that node, and those conditions will generally not be consistent when cycling around a cell.

A procedure destroying node axes is shown by Figure 3.14: the walls of hexagonal cells are moved outward and inward in an alternating way, defining a *reciprocal structure*, cf. [31]. This structure is still formally a honeycomb, if we allow T-junctions.

**Shading Systems.** Wang et al. [19] have proposed shading systems as an application of discrete line congruences. It is in fact not difficult to optimize honeycomb structures so that they serve the same purpose, such that the walls of cells block light effectively while the depth of the honeycomb remains small. We define a vector  $\mathbf{l}$  which corresponds to the direction of light, e.g. at 1 p.m. in summer. Since the angles between walls are always  $120^\circ$  it would not make sense to optimize the honeycomb such that walls are orthogonal to  $\mathbf{l}$ . Instead we look for any field of unit vectors  $\mathbf{a}_i$  attached to the individual vertices  $\mathbf{v}_i$  which are orthogonal to  $\mathbf{l}$  and which serve as prescribed directions of node axes. We then optimize the honeycomb with constraints  $\mathbf{a}_i^T \mathbf{n}_{ij} = 0$  (for all edges  $\mathbf{v}_i \mathbf{v}_j$ ). In the optimization these constraints are given a lower weight than those expressing consistency and angles. A result can be seen in Figure 3.16.

## 3.5 Discussion

The success of the regularized Newton method used here has already been demonstrated by [29]. Referring to that paper, we mention that it works well for constraints which are linear or quadratic, and which do not involve many variables. Similar to [29] we obtain high accuracy. Figure ?? gives details on optimization and geometric properties of the results. Timings refer to an Intel Xeon CPU with 2.67GHz (we also mention that as a sparse linear solver we used the TAUCS library).

*Limitations.* The main geometric limitation of honeycombs has been discussed in §4.3: it is the property that  $120^\circ$  angles between walls imply that the honeycomb's node axes do not in general follow the surface normals of a given reference shape. If we do not insist on this condition (which we could do only for developable surfaces), honeycombs have rather many degrees of freedom, and we found no obstructions in our numerical experiments.

From the viewpoint of statics, structures based on hexagonal meshes are of course more difficult than structures based on triangle meshes, for the simple reason that there are fewer load-bearing edges. In practice, auxiliary force-transmitting elements like cables are typically used. We do not consider this aspect there.

## 3.6 Conclusion

We have presented geometry and computation of honeycomb structures and their applicability to various tasks in freeform architectural design. Notably honeycombs provide a torsion-free support structure with identical nodes, but can be used even for quadrangulation.

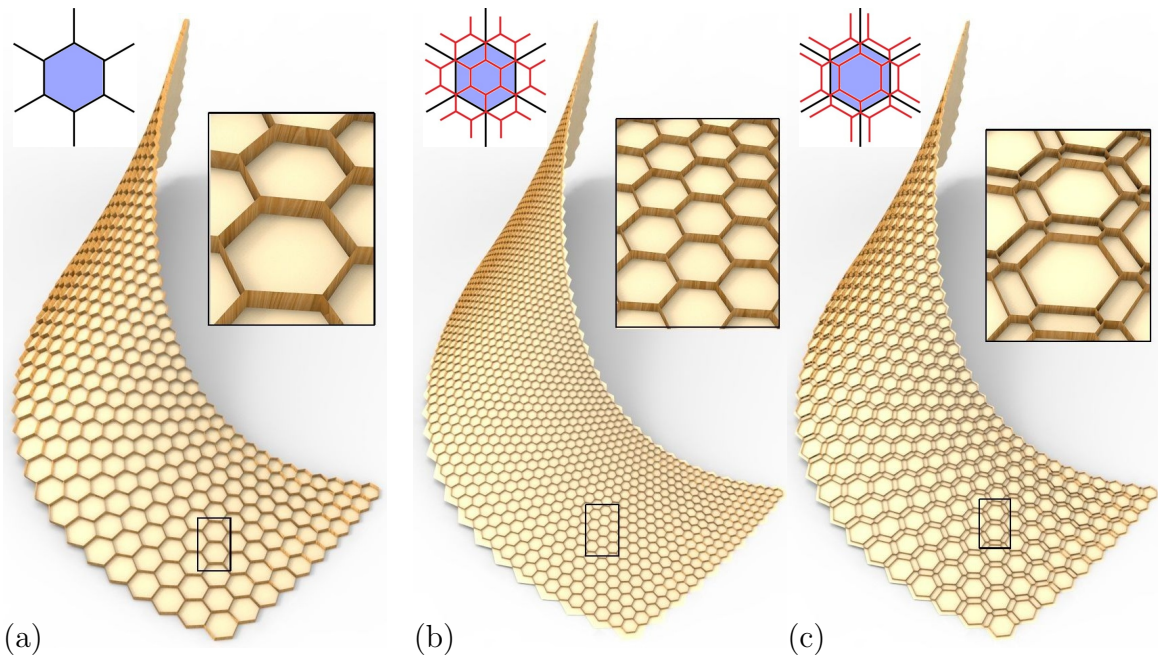


Figure 3.13: Subdivision of honeycombs by graphical stencils. There are several ways we can parallel translate walls of a honeycomb so that intersection properties are maintained, and which are useful for refinement and the definition of other derived structures. These modifications are encoded in a symbolic way by a sketch of a cell or node which shows the old and new positions of walls. (a), (b) The symbol  $\text{[hexagon with 6 lines]} \rightarrow \text{[hexagon with 12 lines]}$  signifies subdivision, generating a regular honeycomb with 1 new cell per old cell, and 1 new cell per old wall, with all new cells roughly the same size. (c) The analogous symbol  $\text{[hexagon with 6 lines]} \rightarrow \text{[hexagon with 12 lines]}$  signifies subdivision which is combinatorially the same but geometrically different, where the newly created cells are of different sizes.

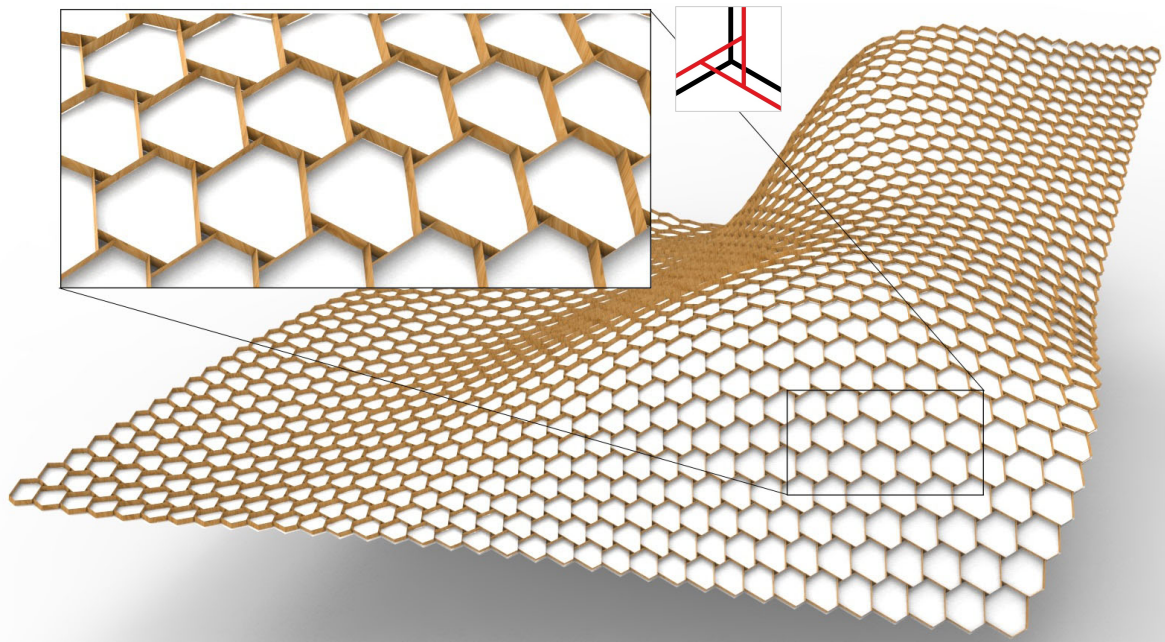


Figure 3.14: Using the language of Figure 3.13, the symbol  $\text{[node]} \rightarrow \text{[triangular cell]}$  signifies the creation of a reciprocal structure, replacing each node by a triangular cell.

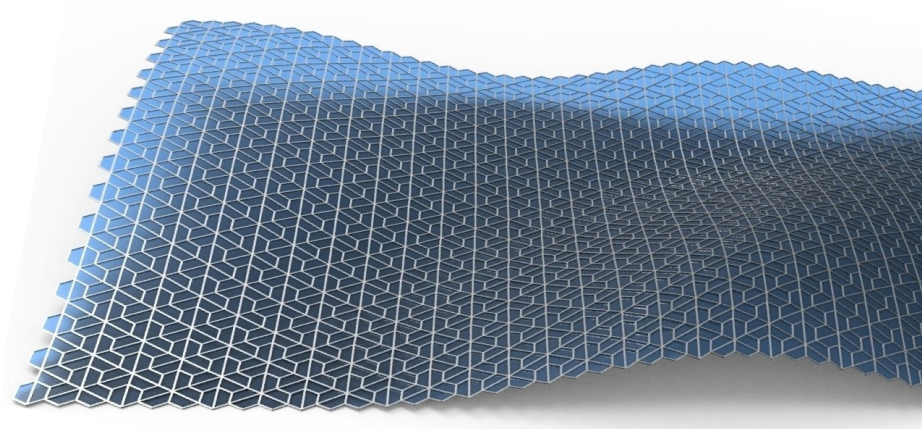


Figure 3.15: A pattern of planar quadrilaterals derived from a honeycomb structure in much the same manner as Figure 3.10, but using a different set of diagonals through cells.

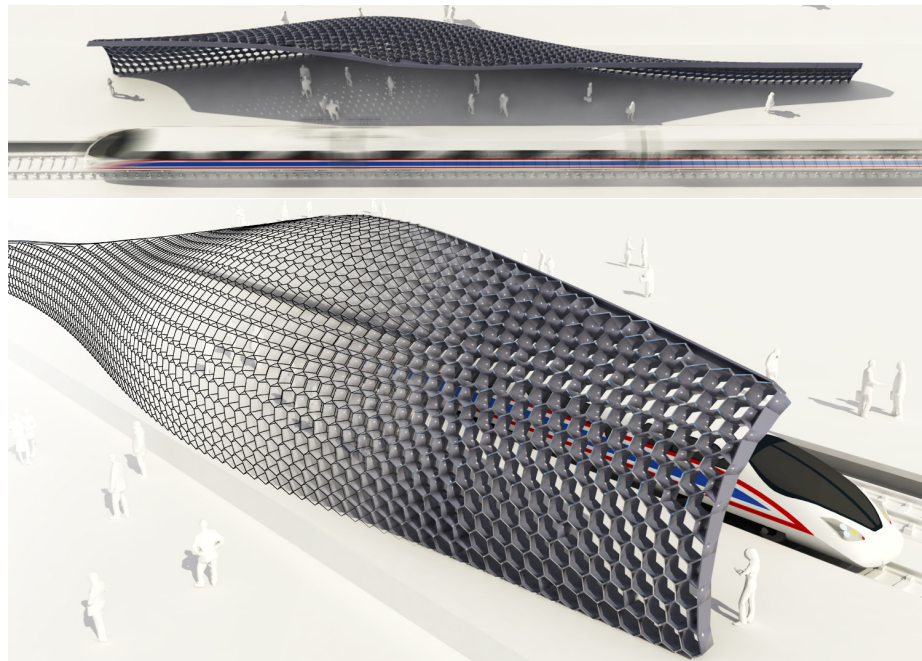


Figure 3.16: Hexagonal support structure as shading system. Given a field of preferred node axes which are orthogonal to the rays of light, we optimize a honeycomb such that the walls of cells fit the given node axis field, thus effectively blocking light while still forming a shallow honeycomb (the result of optimization here serves as freeform structure providing shade for people waiting for trains).



# Chapter 4

## Polyhedral Patterns

### 4.1 Introduction

Architects and engineers are constantly pushing design boundaries by exploring new building shapes and modeling their appearances. Advances in architectural geometry have made it possible for many buildings to be shaped as freeform surfaces. To conform to construction constraints, such designs are often rationalized with *meshes* that have planar faces. These faces are then realized with common materials, such as wood (see Fig. 4.2) or glass.

Symmetric tessellation patterns have often been used in art, architecture, and product design for their aesthetic merits. However, the use of these patterns was restricted to planar surfaces, such as windows, walls, or floors. Notable examples are Arabesques, stained-glass patterns, and mosaics [32, 33]. Here, we seek to enrich architectural design by meshing freeform surfaces with tessellation patterns. Examples of those are in Figures 1.3 and 4.1.

There are several key challenges in designing polyhedral patterns on curved freeform surfaces. Simply placing a given pattern on such a surface (e.g., by a parametrization) and optimizing for tile planarity without any regularization of tile shapes is bound to fail; planarity alone is severely underconstrained, and the process is apt to degenerate to solutions with zero-length edges or foldovers (see Figure 4.3). More-

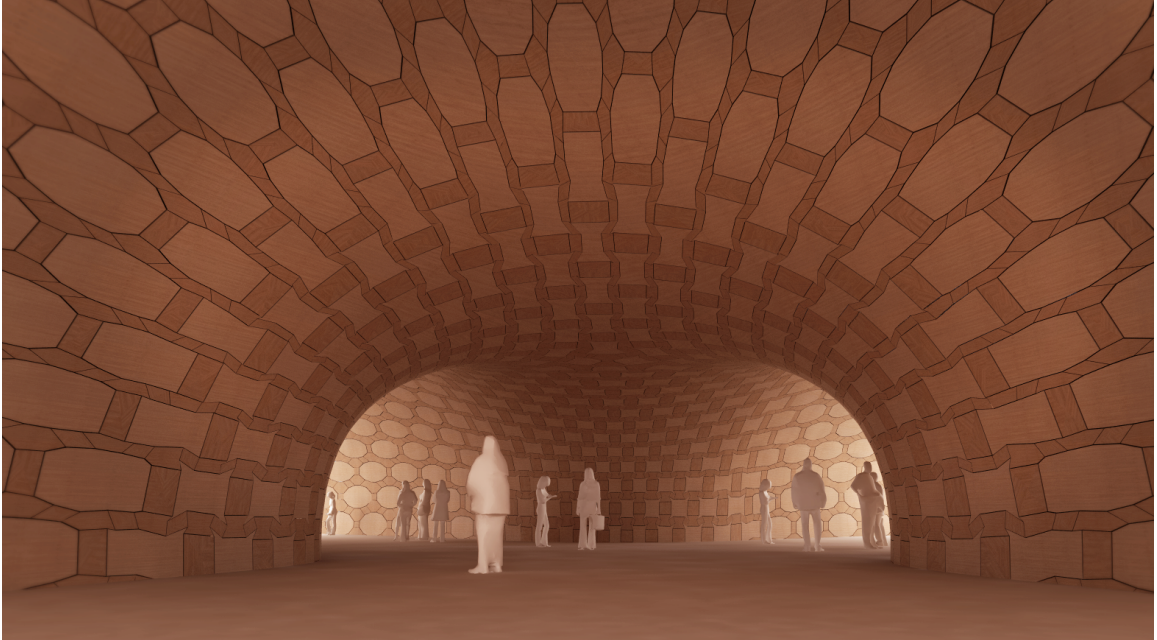


Figure 4.1: Cladding an interior space with a polyhedral pattern using wooden panels. The pattern transforms smoothly from positive to negative curvature regions.

over, an arbitrary choice of tile-shape regularization with commonly used measures, such as face or angle distortion, Laplacian smoothness, or edge-length preservation, might clash with the planarity constraint, resulting in over-constrained optimization. As a consequence, either the mesh is not planarized, or the desired regularity is not satisfied. (Figures 4.26 and 4.27 show examples.)

Our solution to the problem is to study explicit constructions of polyhedral patterns that approximate surfaces with varying Gaussian curvature. We observe curvature-invariant regularities, namely different types of symmetries. We introduce a theoretical study of polyhedral patterns that explains our choice of regularizers, and which leads to an objective function that is neither over- nor under-constrained. We show a set of results that demonstrate, for the first time, the computation of such patterns on surfaces that satisfy both planarity and regularity constraints. We focus on *semi-regular patterns* (see Figure 4.4), which are patterns comprising regular polygons.

Our contributions are:



Figure 4.2: A planar-hexagonal pavilion constructed with wooden panels [34].

- An analysis of surface approximation with polyhedral patterns by describing tile shape deformations, in order to accommodate for curvature. Consequently, we show how different *strip decompositions* result in a variety of patterns.
- Affine symmetries that are curvature-invariant. We construct a family of *regularizers* encoding such symmetries: e.g., symmetries with respect to axes passing through vertices, edge midpoints, or face barycenters, and reflective symmetries with respect to planes.
- A variety of polyhedral patterns that have not been demonstrated before.

## 4.2 Previous Work

Approximation with polyhedral meshes can be achieved with variational shape approximation [35]. However, the resulting unstructured mesh is built to satisfy required approximation accuracy and does not follow a prescribed pattern.

Most previous works that focused on polyhedral mesh creation targeted *planar quad* (PQ) meshes. PQ meshes play a central role in discrete differential geometry [36, 37], and have attracted considerable interest in recent years, cf. [15, 30, 38], as their design is a core problem in architectural geometry.

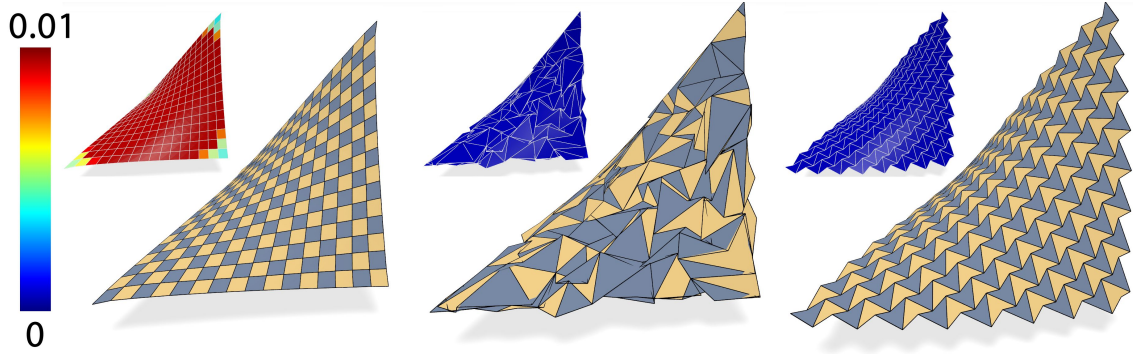


Figure 4.3: Under- and over-constrained optimization illustrated on a quad mesh. The initial mesh is aligned with the parameter lines on a bilinear surface. The task is to planarize it. Left: Using mesh polyline smoothness as a regularizer results in an over-constrained problem, and acceptable planarity is not achieved (red color). Middle: Dropping the regularizer leads to an under-constrained problem, where the faces are perfectly planar, yet their appearance is chaotic and unaesthetic. Right: Using our regularizer, based on affine symmetries (with respect to edge midpoints), yields an aesthetically pleasing pattern with planar quads.

Planar hexagonal (PH) meshes have also been studied, but to a lesser extent. The simplest way to produce them is by taking the dual of triangle meshes [39, 40] or remeshing triangle meshes by parametrization and deformation [41]. They are problematic in several aspects: the face shapes have to change considerably and even become concave in negatively curved areas. In addition, they have to transition smoothly between regions of negative and positive Gaussian curvature. We present a systematic way to regularize patterns between different curvature regions, including PH meshes as a special case.

Special polyhedral patterns appear as by-products with circle-packing meshes [20] and special hexagonal support structures [42]. However, these papers do not consider polyhedral patterns in a systematic way; aesthetics and regularity largely come from the structures the patterns have been derived from.

Numerical optimization schemes for computing polyhedral surfaces include the alternating least-squares approach of [43], the local-global projection method of [44], the augmented Lagrangian algorithm of [45], and the guided projection method of [29].

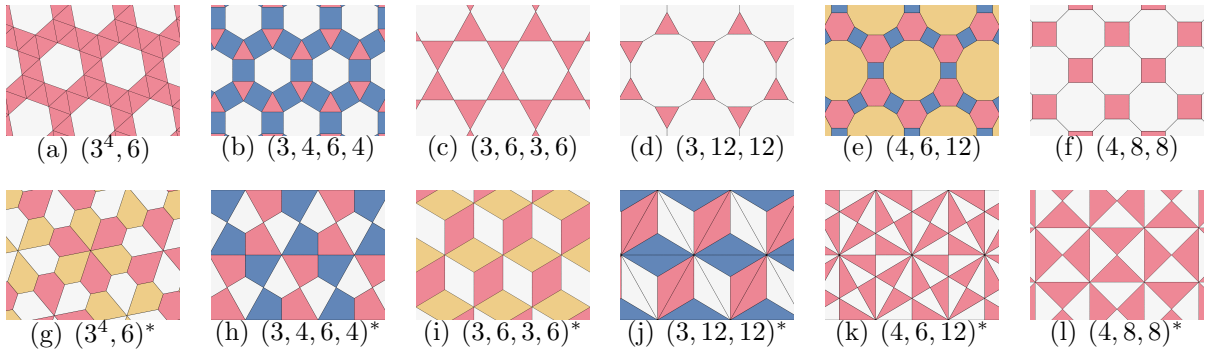


Figure 4.4: Several patterns used in this chapter: selected semi-regular patterns (top row; labels correspond to the valences of faces around a vertex) and their duals (bottom row). In our results, patterns (a) - (e) and (g) - (j) are derived from a hex-mesh, patterns (f) and (l) from a quad mesh, and pattern (k) from a triangle mesh.

Our computations are based on the latter approach, but are distinct from all the other aforementioned works by the novel use of local affine symmetries as regularizers, which are able to adapt to freeform geometry, and by the study of the curvature-dependent appearance of polyhedral patterns.

Triangle meshes are trivial polyhedral meshes. Regularizing them for face and edge repetitivity is the aim of [23] and [46]. These meshes also appear in triangle-based point folding structures [26]. Our symmetry-based regularizers can also be used for triangle mesh optimization.

### 4.3 Geometry of Polyhedral Patterns

Creating polyhedral patterns first and foremost poses a theoretical challenge, since we do not possess the knowledge of how such patterns behave in different curvature regions. We have an understanding of planar-quad meshes as given by [38]. If the network of polylines that is characteristic of quad meshes follows conjugate directions, it is possible to achieve a mesh with smooth polylines. Unfortunately, this is not possible for any orientation of quads (see auxiliary material for a theoretical proof), and there has been no suggestion for what could be done in this case. An analysis is

provided for feasible planar hexagonal tile shapes by [40]. However, the description is particular for hexagons in principal directions, and the generalization to semi-regular patterns is not obvious.

In the following, we provide an analysis of feasible planar tile shapes in different curvature regions, for the general case of semi-regular tilings. In Section 4.6, we utilize the insights gained from this analysis, to establish a set of symmetries that remain invariant in each curvature region. We consequently use these symmetries as regularizers in our planarization algorithm.

We base our geometric constructions on *semi-regular patterns*, which are tilings that can be derived in the plane by altering either of the three regular tilings: triangle, square, or hexagonal grids. Semi-regular tilings are characterized by several properties: First, the neighborhood of any vertex is perfectly similar to the neighborhood of any other. Second, such tiles constitute an *orthogonal circle pattern*: every tile has a circumcircle, and the dual segments between neighboring tile (circle) centers are orthogonal to the primal edge they share. This property is important when we discuss construction by lifting. We depict a range of semi-regular tilings that we employ in Figure 4.4. We denote tilings using *vertex configuration* shorthand by numbering the degree of faces around each vertex and using powers for multiples, e.g., a pure hexagonal pattern is  $6^3$ . We use the term “tile” to indicate a single face of the tiling, and either “tiling” or “pattern” to indicate the entire set. We often refer to the dual pattern as the pattern that is made by connecting dual face centers of adjacent primal tiles. Tilings that are not semi-regular exhibit several of these properties, and our consequent optimization results in interesting polyhedral patterns as well.

### 4.3.1 Discretization of osculating paraboloids

We construct some explicit embeddings of polyhedral patterns on surfaces to study necessary deformations in tile shapes. Such deformations are the result of fitting

planar patterns with given connectivities onto curved surfaces, while constraining each tile to remain planar. Our purpose is to derive the invariants of the required deformations, focusing on symmetries they fix. An understanding of such invariant symmetries serves as a guide to predicting the resulting pattern shapes expected within our optimization process.

We locally approximate the original surface,  $S$ , to a second-order in a point,  $\mathbf{p} \in S$ , with an osculating paraboloid,  $S_2$ . Assuming the  $z$  direction is parametrized to be in the direction of the normal, the formula defining the paraboloid is  $2z = \kappa_1 x^2 + \kappa_2 y^2$ , where  $\kappa_1, \kappa_2$  are the principal curvatures, and the  $x$  and  $y$  axes are the respective principal directions. The paraboloids are characterized as either elliptic (both curvatures are nonzero and have the same sign), hyperbolic (different nonzero signs), or cylindrical (one of the curvatures is zero). In case that  $\kappa_1 = \kappa_2 = 0$ , the osculating paraboloid is a plane, and we do not need to deal further with this trivial case.

We consider discretizations of paraboloids by polyhedral patterns characterized by two properties: First, they are *inscribed*, which means the vertices of the pattern lie exactly on the paraboloid. Second, we have *normal adherence*. Assuming that the supporting plane to the inscribed tile encloses a well-defined small patch of the paraboloid, then there must be a point within the patch whose tangent plane is parallel to the supporting plane. Both properties can be generally relaxed, but it is cogent to study the pattern symmetry and regularity emerging from these most restrictive requirements.

**Lifting** The analysis we give for tiling surfaces relies on lifting planar tiles onto paraboloids (see Figure 4.6). We then explore the tile shapes and topologies for which the lifting produces polyhedral patterns (preserving tile planarity). Given the paraboloid  $S_2$ , the vertices are lifted (bijectively) by the function  $(x, y) \rightarrow (x, y, \kappa_1 x^2 + \kappa_2 y^2)$ . The intersection of the supporting plane of the tile and the paraboloid is a conic, re-

lated to the *Dupin indicatrix*. The projection of the conic down to the plane is again a planar conic, endowed with required properties that we next detail. Moreover, the conic on the paraboloid is an *affine image* of the planar tile. See Figure 4.5 for an example.

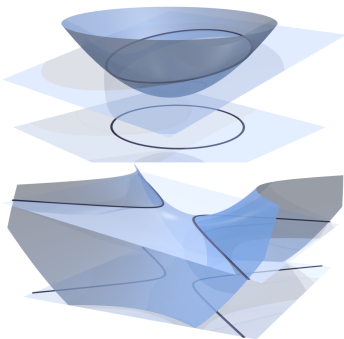


Figure 4.5: Lifting. Top: a circle lifted to an ellipse in a rotational paraboloid. Bottom: a hyperbola lifted to a hyperbolic paraboloid.

**Consistent tilings** Polyhedral patterns on paraboloids are synonymous with consistent structures that are pivotal to our framework and that govern the deformations induced on tiles for consistent approximation of paraboloids. Assume that there are two neighboring tiles,  $i, j$ . Their centers are defined by looking at the centers of the conics on the paraboloid and projecting them down on the tiling plane, producing  $\mathbf{c}_i, \mathbf{c}_j$ . The intersection points between them are the projected common vertices,  $\mathbf{p}_i, \mathbf{p}_j$ . We define the primal vector  $\mathbf{p}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ , and the dual vector  $\mathbf{c}_{ij}$  similarly. Next, we consider the metric induced by the paraboloid:  $\langle \mathbf{a}, \mathbf{b} \rangle := \kappa_1 x_a x_b + \kappa_2 y_a y_b$ . The lift of a planar tiling into a paraboloid is polyhedral if and only if:

- (A). **Conjugacy:** the primal and dual vectors are *conjugate*. i.e., they are orthogonal with respect to the metric ( $\langle \mathbf{p}_{ij}, \mathbf{c}_{ij} \rangle = 0$ ).
- (B). **Bisection:** The dual edge bisects the primal edge.

We call a tiling that obeys both properties *consistent*. For completeness, we include a proof in the Appendix. Consistency brings about several key consequences:

- If the paraboloid is a rotational-symmetric canonical paraboloid  $2z = x^2 + y^2$ ,



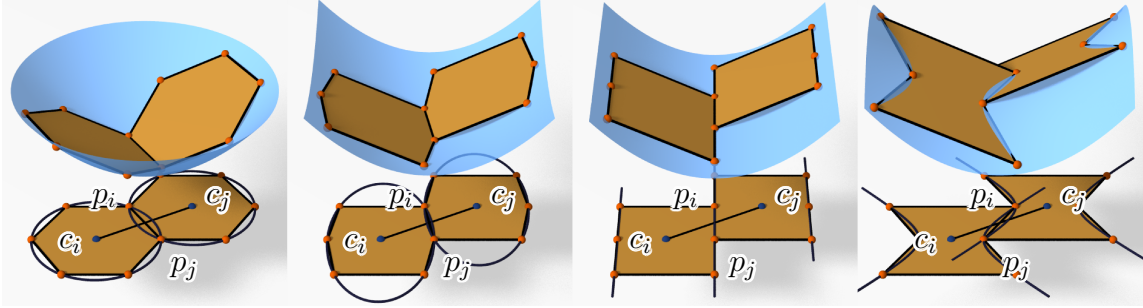


Figure 4.6: Lifting consistent tilings to paraboloids. Under the induced metric in the plane, the dual edge  $\mathbf{c}_{ij}$  and the primal edge  $\mathbf{p}_{ij}$  are conjugate. Both vectors are orthogonal in the rotational paraboloid case (left).

then the duals and the primals are in fact also Euclidean orthogonal, as we demand from the original tilings.

- If the paraboloid is cylindrical, then either the dual or the primal must be in the direction of the *ruling* (the direction of zero curvature). That means that the tiling has to comprise strips of faces that are parallel to the ruling direction.
- If the paraboloid is hyperbolic, the dual and the primal can be identical (in *asymptotic* directions). We can potentially produce degenerate and concave tiles in this manner.

### 4.3.2 Fitting tiles to paraboloids

Given a planar tiling and a paraboloid of any shape, we next wish to deform the tiling on the plane, such that the lifting produces a polyhedral pattern. This should be done while keeping the shapes of the tiles as symmetric and as regular as possible and with minimal deformation. It is worth noting at this point that we use this construction for a theoretical, rather than a direct algorithmic, purpose; it is a general and arbitrary way to observe which symmetries are invariant within the deformation, and to motivate our use of such symmetries in pattern optimization for general surfaces, locally approximated as paraboloids. We initiate this analysis by considering the

canonical rotational paraboloid  $S_r : 2z = x^2 + y^2$ . We take any semi-regular tiling on the plane. Without any deformation, the circular faces of the semi-regular tiling are then projected into ellipses in the paraboloid, which are planar by definition. We thus obtain a natural polyhedral pattern embedded in  $S_r$  for every semi-regular pattern.

**Tiling cylindrical paraboloids** Without loss of generality, we assume that our cylinder is defined by  $S_c : z = \kappa_2 y^2, \kappa_2 > 0$ . By the rule of conjugacy, the tiling must comprise strips that are parallel to the ruling direction,  $\hat{\mathbf{y}}$ , to consistently tile the cylinder. In light of this, and opting to deform the tiling as little and as symmetrically as possible, we do the following: decompose the tiling into strips of faces that are parallel to the rulings. The lines between dual conic centers of the same strips are denoted as *dual rulings*, and the sequence of intersection edges bounding two strips are denoted as *primal rulings*. Next, fix all the tile (dual) centers and deform all (primal) vertices orthogonally to the ruling until they are on a parallel primal ruling. The actual position of the primal ruling is set as the ruling that is closest to the original primal vertices (see Figure 4.7 for a depiction of this process). Denote the original position of any primal vertex as  $\mathbf{p}_i$ , and the deformation vector for a cylindrical pattern as  $\mathbf{u}_{c,i}$ . Then,  $\hat{\mathbf{y}} \cdot \mathbf{u}_{c,i} = 0$ .

**Tiling elliptic and hyperbolic paraboloids** To unify our deformation setting and make it fit all types of paraboloids, we rephrase our construction for a cylindrical paraboloid in a local and continuous manner: we deform the primal edges of an initial tiling so that the (constant) dual and the primal are conjugate according to the induced metric, and the deformation is done according to the choice of strip decomposition, as in the cylindrical case. Suppose again that the chosen direction of the dual strip is  $\hat{\mathbf{y}}$ . We then need to find the deformation vector  $\mathbf{u}_i$  of vertex  $\mathbf{p}_i$  so that for each primal edge  $\mathbf{p}_{ij}$  and dual edge  $\mathbf{c}_{ij}$  we get  $\langle \mathbf{p}_j + \mathbf{u}_j - \mathbf{p}_i - \mathbf{u}_i, \mathbf{c}_{ij} \rangle = 0$ , according to the metric. Furthermore, we constrain  $\mathbf{u}_i = -\mathbf{u}_j$  for symmetry. Since  $\mathbf{u}_i = -\mathbf{u}_j$  are orthogonal to  $\hat{\mathbf{y}}$ , it is straightforward to compute the actual deformation.

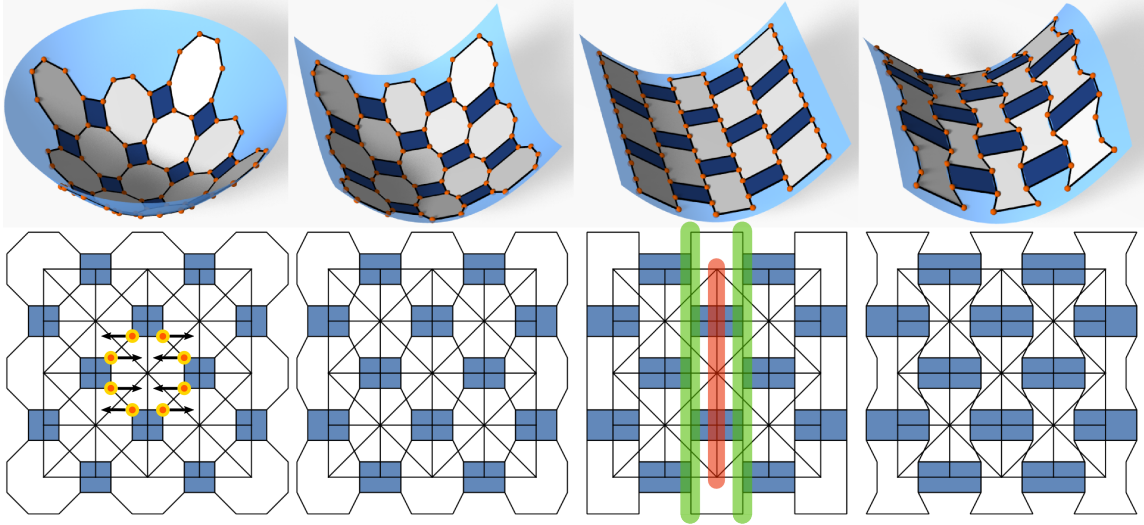


Figure 4.7: Deforming and lifting. Top: the paraboloid tilings. Bottom: top view of the tiling. Left to right: original (fit to the canonical paraboloid), anisotropic elliptic, cylindrical, and hyperbolic.

**Discussion** Our deformation process is obviously invariant to scale. More accurately, it only pertains to the ratio of  $\kappa_1$  and  $\kappa_2$  and not their sizes. In addition, it is evident that tiles with more than 4 vertices must become non-convex in order to be inscribed in negatively curved regions; this is expected, since the conic in which the tile is inscribed is a hyperbola.

The deformation orthogonal to the chosen strip direction is the only one possible for tilings such as  $6^3$ . However, by constraining the consistency, and by conforming to the aesthetic request that the repeating faces of the same type must stay congruent, some tilings may in fact allow more degrees of freedom in deformation possibilities. This works only to our advantage.

**Violating consistency** The deformations we defined maintain conjugacy and bisection for most semi-regular patterns, but not for some. For example, consider the  $(4, 6, 12)$  example in Figure 4.8: by fixing all the dual vertices, our deformation would violate bisection between some of the quads and the 12-sided faces. This is caused by the special structure of the  $(4, 6, 12)$  pattern, for which the dual centers of the hexes and the oblique quads cannot conform to straight rulings should they stay fixed. Our

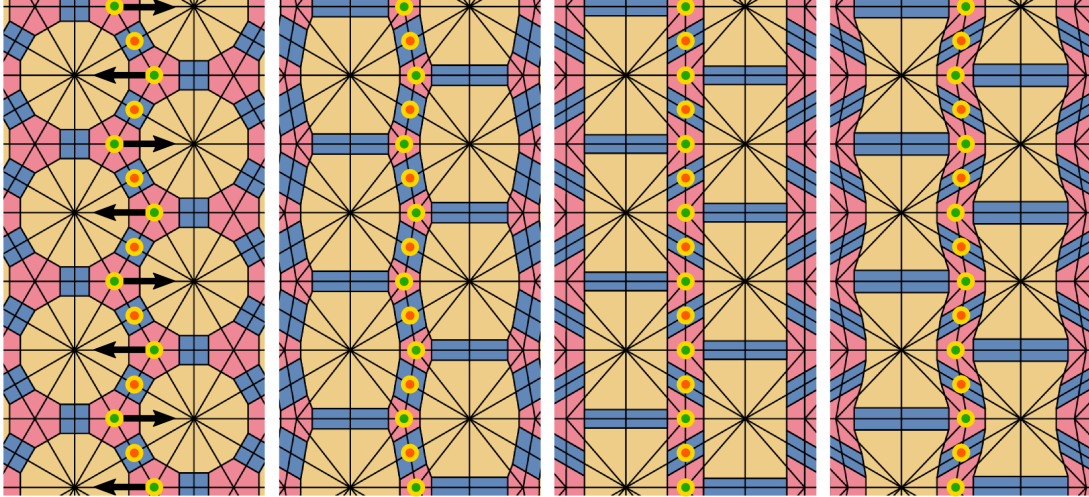


Figure 4.8: The dual centers of the  $(4, 6, 12)$  pattern do not correspond to possible rulings if fixed upon deformation. Allowing the hex (pink) centers to deform fixes this problem.

correction is simple: allow the centers of the hexes to deform as well, so that they line up with the (fixed) centers of the oblique quads.

### 4.3.3 Strip decompositions

Our explicit construction provides a canonical way to approximate paraboloids, by relying on a single possible choice for defining strips. In the following, we explore other possible constructed solutions, by choosing different alignments, corresponding

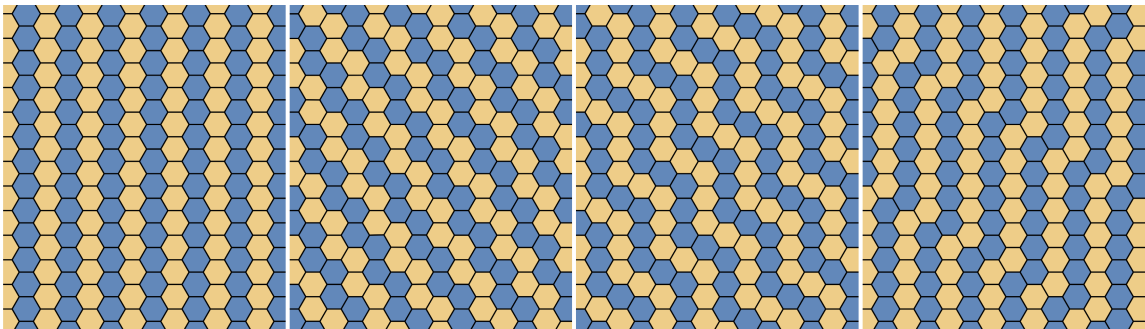


Figure 4.9: Different strip decompositions for regular hexagons. The three decompositions from the left correspond to the ones shown in Figure 4.10. In addition, the transformation corresponding to the decomposition second from the left is shown in Figure 4.11.

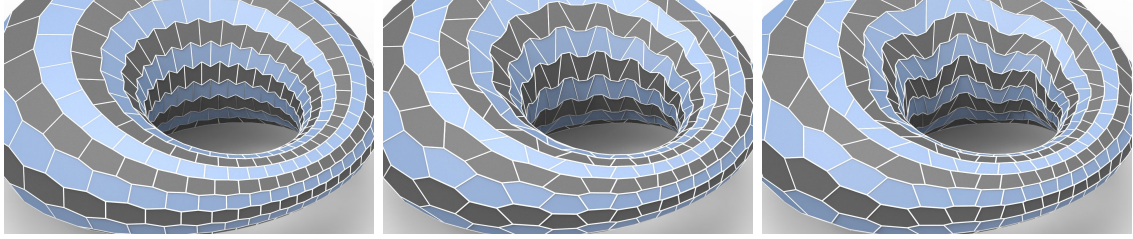


Figure 4.10: Different strip decompositions increase the available types of polyhedral patterns. With the three strip decompositions shown in Figure 4.9, we obtain hexagon patterns approximating a cyclide with different appearances.

to decomposing the patterns into different strips.

**Decomposable patterns** Essentially, strip decomposition is a combinatorial refinement of the original pattern. A feasible strip decomposition is a collection of disjoint dual strings (trees with 2-valence nodes; see Figure 4.9 for examples). Since a strip decomposition assigns primal vertices to dual vertices, it is actually a strip decomposition of the dual pattern as well. Regular quads, hexagons,  $(3^4, 6)$ ,  $(3, 4, 6, 4)$ ,  $(4, 6, 12)$ ,  $(4, 8, 8)$  and their dual patterns have infinitely many strip decompositions. However, patterns such as  $(3, 6, 3, 6)$ ,  $(3, 12, 12)$  and their duals cannot be decomposed to strips by definition.

**Deformations and symmetries** The practical meaning of choosing strips is to contrast a chosen strip direction with the principal directions of the paraboloid. Choosing a strip means choosing a dual axis for every consequent pair of faces, and forcing the primal vertices to move in directions that are orthogonal to the strip axis. In other words, we constrain a *plane of symmetry* that is orthogonal to the tile, and passes through its center.

However, as we explain above, the principal directions may mandate (e.g., if they are rulings), that the surface forms lines of primal vertices along such rulings. At any rate, the tile must also be inscribed to a conic of the same nature. The *canonical* choice of strips is where the dual axis is aligned with the rulings (see Figure 4.9 left). However, other choices may produce interesting patterns due to the mismatch

between the constrained symmetry and the rulings (see Figure 4.11), and they may potentially form invalid configurations. Such configurations arise when the strips are along the asymptotic directions of the surface, i.e., where the dual direction is self conjugate.

**General patterns** Not all patterns can be decomposed to strips. For instance, the tri-hex pattern  $(3,6,3,6)$  cannot be decomposed. Such patterns cannot therefore comply to the normal adherence and cannot be made consistent by deformation. However, as our algorithm requires consistency only for the theoretical analysis, we still utilize these patterns in practice, just without any guarantees. Figures 4.16 and 4.22 provide examples.

#### 4.3.4 Regularizers Motivated by Symmetries

Ideally, we would like to achieve the described symmetric and planar tile shapes for meshes initially tiled with semi-regular convex patterns. However, general meshes are not paraboloids, and they have a variety of strip decompositions and varying curvature regions. Not wanting to be particular for every pattern, we instead opt for the most general way to make any type of semi-regular pattern deform properly. Our point is to utilize what *remains invariant* under curvature-based tile deformations, rather than what *deforms*. Therefore, we identify invariant symmetries of tiles and

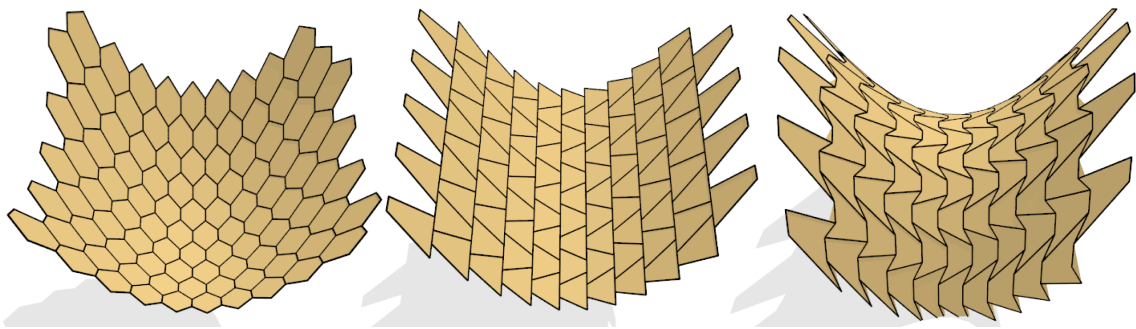


Figure 4.11: Transformation of the regular hexagon pattern from a rotational paraboloid (left) via a parabolic cylinder (middle) to a hyperbolic paraboloid (right) with the strip decomposition shown in Figure 4.9, second from left.

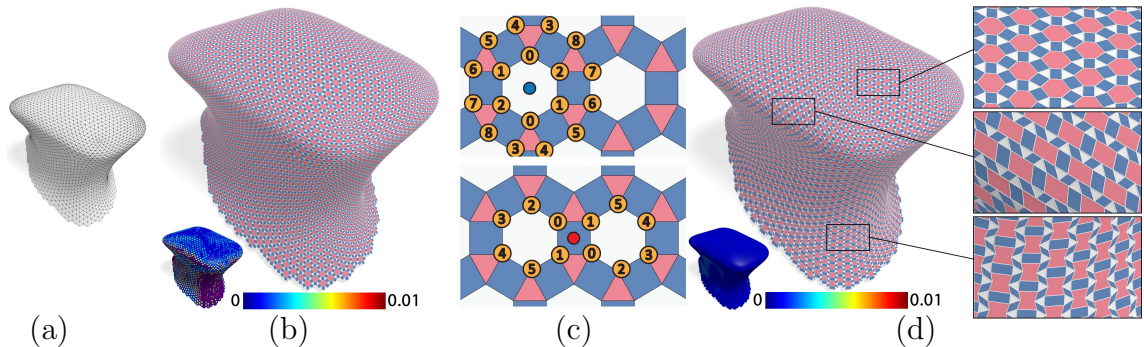


Figure 4.12: Framework overview: a) for an initial triangle, quad, or hex mesh, we can generate a pattern mesh using simple geometric rules. b) The initial pattern mesh might already be aesthetically pleasing, but the faces are typically not planar. c) A regularizer can be configured by specifying symmetries that should be preserved in the pattern. In this case, face symmetries are chosen. Corresponding vertex pairs are shown using the same number and the symmetry centers are shown in blue and red. d) Finally, the optimization generates a mesh with planar faces. The most interesting aspect of polyhedral patterns is that most of them have to transform so that they look different in regions of positive, zero, and negative Gaussian curvature (see insets).

then regularize the mesh in our planarization process to maintain them. The symmetries that we identify include reflection through axes and through planes as well as reflections through the centers of tiles or edges. It is straightforward to check that such symmetries are general enough to contain the deformations we describe here. The symmetries are described in greater detail in Section 4.6.

## 4.4 Overview and User Interaction

Our framework comprises four stages, shown in Figure 4.12.

**Pre-processing** Initial meshes are generated using triangular-, quad-, or hex-based remeshing techniques in a separate program, according to the desired pattern (see Figure 4.4 for a description). Many patterns are initialized using the hex-based remeshing approach proposed by Vaxman and Ben-Chen [41] with their planarity optimization omitted. We therefore have two input meshes in our system: A finely-tessellated triangle mesh to define the reference surface and a coarser (non-planar) remeshed

triangle, quad, or hex mesh.

**Pattern generation** The user can transform the initial coarse mesh into a pattern mesh by selecting from a list of pre-defined patterns. The transformation is implemented using a sequence of geometric rules, e.g., subdivision rules. The implementation of such rules is fairly straightforward and follows the framework proposed by Akleman et al. [47].

**Symmetry configuration** The user can then specify a desired strip decomposition and configure the regularizer by assigning symmetries. We offer axial symmetries with respect to an axis passing through a vertex, an edge midpoint, or a face barycenter. We also offer reflective symmetries with respect to a plane, e.g., a plane passing through an edge. The user specifies the symmetry assignment for one or more elements, and the system propagates the assignment over the whole mesh according to the strip decomposition. To guide the user in his/her selection, we provide a list of suggested symmetries. The suggestions are generated by mapping each strip decomposition of each pattern to a cylinder and then observing what symmetries are feasible (see Sec. 4.3).

**Symmetry optimization** Our algorithm optimizes the pattern for planarity and aesthetics (using the regularizer configured in the previous step) through non-linear optimization. The details for the optimization framework are presented in Section 5.4, and those for the symmetry regularizers are given in Section 4.6.

## 4.5 Optimization Framework

We next describe the regularity-based planarity optimization framework that our work builds upon. The inputs are a reference surface,  $S$ , given as a triangle mesh, and an initial polygonal mesh with vertices,  $\mathbf{v}_i$ , that approximates the reference surface. The goal is to optimize the initial polygonal mesh,  $M = (V, E, F)$ , according to three



terms: the planarity of the faces, the closeness to the reference surface, and the regularity of the mesh. We rely on existing methods (described in this section) to formulate planarity and closeness terms. The regularity terms are our contribution.

**Variables** We denote the vertex coordinates of  $M$  as  $\mathbf{v}_i$ ,  $i \in V$ , and the unit face normals as  $\mathbf{n}_k$ ,  $k \in F$ . Vertices are not constrained to lie on the reference surface,  $S$ , exactly. The closest point on  $S$  for a vertex  $\mathbf{v}_i$  is  $\mathbf{v}_i^*$  with corresponding normal  $\mathbf{n}_i^*$  (see Fig. 4.13).

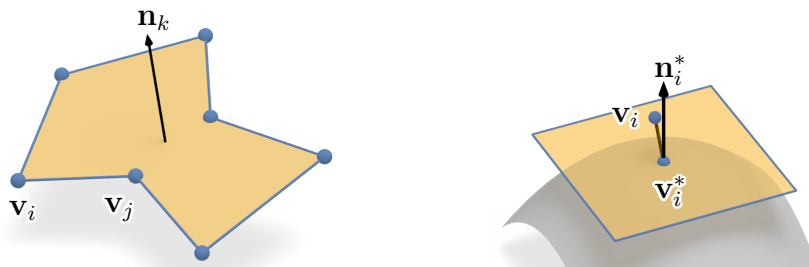


Figure 4.13: Notation: faces (left), closest point projection (right)

**Problem formulation** The objective function we minimize is

$$E(\mathbf{v}_i) = \lambda_1 E_{plan} + \lambda_2 E_{close} + \sum_j \mu_j E_{reg}^j. \quad (4.1)$$

Following the reasoning of [29], we set up a system with energies that are at most quartic, which entails soft constraints that are at most quadratic, since this formulation is easy to optimize using a standard regularized Gauss-Newton algorithm.

**Planarity** The planarity constraint is necessary for all non-triangular faces. We adapt the formulation of [29] and express  $E_{plan}$  as

$$E_{plan} = \sum_{k \in F} \sum_{(i,j) \in E(f_k)} ((\mathbf{v}_i - \mathbf{v}_j) \cdot \mathbf{n}_k)^2 + \sum_k (\mathbf{n}_k \cdot \mathbf{n}_k - 1)^2, \quad (4.2)$$

which is zero if all face edges are orthogonal to a unit length normal.

**Closeness** The closeness constraint of a vertex,  $\mathbf{v}_i$ , to a reference surface is modeled by requiring  $\mathbf{v}_i$  to move only on the tangent plane associated with its closest point,

$\mathbf{v}_i^*$ , on the reference surface,  $S$ :

$$E_{close} = \sum_{\mathbf{v}_i \in V} ((\mathbf{v}_i - \mathbf{v}_i^*) \cdot \mathbf{n}_i^*)^2. \quad (4.3)$$

As shown in Figure 4.13,  $\mathbf{n}_i^*$  is the normal of the tangent plane at  $\mathbf{v}_i^*$ , and it is kept constant in every iteration. Alternatively, for coarse and inconsistent tilings, we may use closeness of face barycenters instead to relax this constraint (see Figures 4.17 and 4.28, left).

**Previous iteration** We add a term in each iteration that dampens the optimization for stability by closeness to the previous iteration:

$$E_{prev} = \beta \sum_{\mathbf{v}_i \in V} |\mathbf{v}_i^m - \mathbf{v}_i^{m-1}|^2 + \beta \sum_{\mathbf{n}_i \in F} |\mathbf{n}_i^m - \mathbf{n}_i^{m-1}|^2, \quad (4.4)$$

where  $m$  denotes the number of an iteration and  $v_i^m$  is the value of  $v_i$  at iteration  $m$ . We use  $\beta = 10^{-6}$  in all our examples.

## 4.6 Regularization with Affine Symmetries

We next define the invariant symmetries of deforming patterns (see Section 4.3) and how we utilize them in practice to regularize patterns undergoing deformations through the planarization process. Generally speaking, there are several ways to represent feasible regularities, such as enforcing specific angles, polyline smoothness of selected sequences of non-adjacent vertices, ratios of edge lengths, and more. We choose to use local affine symmetries as described in the following, because they are simple, sparse, local and linear (e.g., compared with angle-based formulations), and this is important for computational efficiency.

We describe the practical implementation of various symmetry regularizers and adapt them to the discrete surface by two approaches: affine symmetries in space and

in a tangential projection. Each approach has different merits and shortcomings.

### 4.6.1 Affine symmetries

Affine symmetries can be defined with respect to either an axis or a plane. We can distinguish four different generators of symmetries: vertices, faces, edge midpoints, and edges. Vertices, faces, and edge midpoints generate symmetries with respect to an axis and edges generate symmetries with respect to a plane.

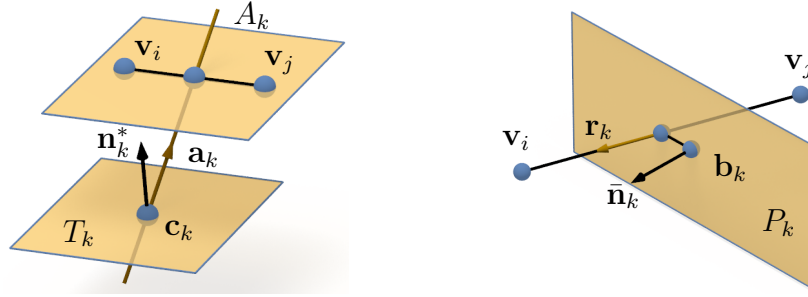


Figure 4.14: Affine reflection in an axis (left) and in a plane (right)

**Axial symmetries** An affine reflection in an axis,  $A_k$ , requires the additional prescription of a reference plane,  $T_k$  (not parallel to  $A_k$ ; see Fig. 4.14, left). Then, a pair of vertices  $\mathbf{v}_i$  and  $\mathbf{v}_j$  is symmetric with respect to  $A_k$  if the midpoint between  $\mathbf{v}_i$  and  $\mathbf{v}_j$  lies on  $A_k$  and the vector  $\mathbf{v}_i - \mathbf{v}_j$  is parallel to  $T_k$ . Let  $A_k$  be defined by a direction vector  $\mathbf{a}_k$  and a point  $\mathbf{c}_k$ , and let  $\mathbf{n}_k^*$  be a normal vector of  $T_k$ . Then, the axial symmetry regularizer  $E_{reg}^1$  is encoded as follows:

$$\sum_{(i,j,k)} \left( \frac{(\mathbf{v}_i + \mathbf{v}_j)}{2} - (\mathbf{c}_k + \lambda_{kl}\mathbf{a}_k) \right)^2 + \left( (\mathbf{v}_i - \mathbf{v}_j) \cdot \mathbf{n}_k^* \right)^2 \quad (4.5)$$

The triplets  $(i, j, k)$  are chosen according to the user-assigned symmetries. That is,  $\mathbf{v}_i$  and  $\mathbf{v}_j$  are selected to have affine symmetry with respect to  $A_k$ . Furthermore,  $\mathbf{a}_k$  and  $\lambda_{kl}$  are considered as additional variables in our optimization. The axis point  $\mathbf{c}_k$  is a vertex, an edge midpoint, or a face barycenter, and thus it is a linear combination of vertex coordinates. The  $\mathbf{n}_k^*$  variables can be either considered as additional

variables, or approximated at the beginning of each iteration as the normal of  $S$  at the closest point to  $\mathbf{c}_k$ . We do the latter. Symmetry is applied to local neighborhoods as illustrated in Fig. 4.15. Note that the symmetry of a planar face with respect to its barycenter does not require an axis.  $\mathbf{n}_k^* = \mathbf{a}_k$  models a Euclidean reflection in  $A_k$  and is therefore suitable to enforce Euclidean symmetries.

**Plane-reflective symmetries** Fig. 4.14 (right) presents an affine reflection with respect to a plane,  $P_k$  (through point  $\mathbf{b}_k$  and with normal vector  $\bar{\mathbf{n}}_k$ ), in the direction  $\mathbf{r}_k$ . Plane-reflective symmetry of  $\mathbf{v}_i$  and  $\mathbf{v}_j$  requires their midpoint to be located on  $P_k$ , and the vector  $\mathbf{v}_i - \mathbf{v}_j$  to be parallel to  $\mathbf{r}_k$ . We encode this requirement in the regularizer,  $E_{reg}^2$ , as follows:

$$\sum_{(i,j,k)} (((\mathbf{v}_i + \mathbf{v}_j)/2 - \mathbf{b}_k) \cdot \bar{\mathbf{n}}_k)^2 + ((\mathbf{v}_i - \mathbf{v}_j) - \lambda_{kl}\mathbf{r}_k)^2. \quad (4.6)$$

We use  $\mathbf{r}_k$  for each reflection plane and the scale variable  $\lambda_{kl}$  as additional variables. The plane  $P_k$  is commonly the bisection plane of two adjacent faces in a polyhedral mesh, and so  $\mathbf{b}_k$  can be the midpoint of the common edge of two adjacent faces. The normal,  $\bar{\mathbf{n}}_k$ , is then pre-estimated in each iteration. With  $\mathbf{r}_k = \bar{\mathbf{n}}_k$ , we obtain a Euclidean reflection.

**Symmetry centers** In Fig. 4.15, as exemplified on a quad mesh, the three leftmost images show point symmetries in 2D, equivalent to 3D axial symmetries. The blue dot represents the symmetry center. Each pair of symmetric points is labelled the same (orange dots). Their symmetry centers are located at a vertex, an edge midpoint or a face barycenter, and their symmetries are denoted accordingly. The rightmost edge symmetry is equivalent to a 3D planar symmetry.

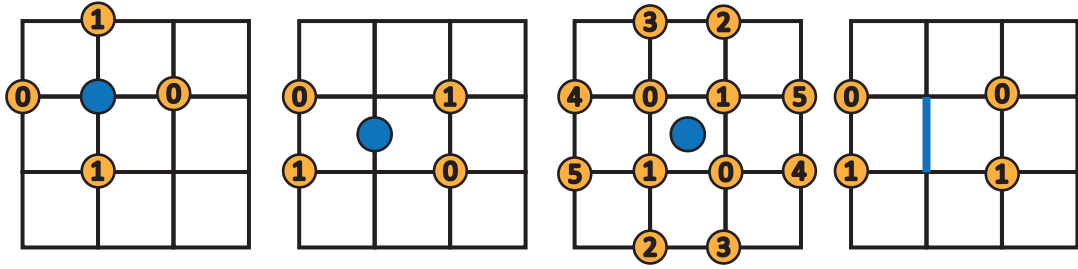


Figure 4.15: Left to right: symmetry with respect to a vertex, an edge midpoint, a face barycenter, and an edge.

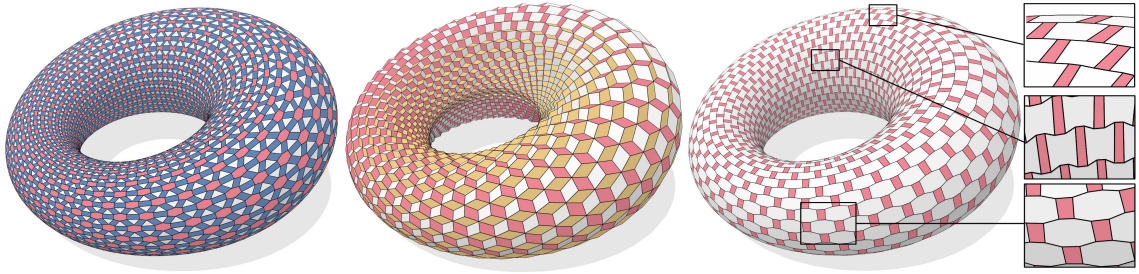


Figure 4.16: Semi-regular patterns on a Dupin cyclide. Left: A  $(3, 4, 6, 4)$  pattern using face symmetries. Middle: A  $(3, 6, 3, 6)^*$  pattern using symmetries with respect to an edge. Right: A  $(4, 8, 8)$  pattern using face symmetries.

## 4.6.2 Symmetry in a tangential projection

A relaxed version of affine axial symmetry is symmetry in a projection parallel to a certain direction (the image plane of the projection does not matter). To achieve it, we simply discard the second part of Equation 4.5. We use the normal at the closest point,  $\mathbf{c}_k^*$ , to  $\mathbf{c}_k$  as the projection direction, and thus enforce a symmetry that is relative to the tangent plane of  $S$  at  $\mathbf{c}_k^*$ . This yields the regularizer  $E_{reg}^3$ :

$$\sum_{(i,j,k)} ((\mathbf{v}_i + \mathbf{v}_j)/2 - (\mathbf{c}_k + \lambda_{kl}\mathbf{a}_k))^2. \quad (4.7)$$

By approximating  $\mathbf{a}_k$  with  $\mathbf{n}_k^*$ , this expression can be simplified to the following equivalent formulation:

$$\sum_{(i,j,k,l)} (((\mathbf{v}_i + \mathbf{v}_j)/2 - \mathbf{c}_k) \cdot \mathbf{t}_{kl})^2. \quad (4.8)$$

In addition, we need to sum over two orthogonal directions,  $\mathbf{t}_{k1}$  and  $\mathbf{t}_{k2}$ . Both directions are orthogonal to  $\mathbf{n}_k^*$ , and they are estimated at the beginning of each iteration.

### 4.6.3 Avoiding self-intersection

The proposed symmetry regularizers cannot prevent self-intersections within the pattern. To counter that, we introduce an additional regularizer,  $E_{reg}^4$ , assuming that a line segment connecting the face barycenter to a vertex is within the corresponding face:

$$\sum_{\substack{k \in F \\ 0 \leq i < |f_k|}} ((\hat{\mathbf{v}}_k^i - \mathbf{c}_k) \times (\hat{\mathbf{v}}_k^{i+1} - \mathbf{c}_k) \cdot \mathbf{n}_k - \nu_{ki}^2)^2. \quad (4.9)$$

We assume that the faces are consistently oriented with vertices  $(\hat{\mathbf{v}}_k^0, \hat{\mathbf{v}}_k^1, \dots, \hat{\mathbf{v}}_k^{|f_k|-1})$ , where indices in the sum are taken modulo the face valence  $|f_k|$ . The face normals,  $\mathbf{n}_k$ , and barycenters,  $\mathbf{c}_k$ , are evaluated prior to each iteration, and considered as constants. We introduce  $\nu_{ki}$  as *slack variables* to encode the inequality requirements that the vectorial areas of  $\mathbf{c}_k \hat{\mathbf{v}}_k^i \hat{\mathbf{v}}_k^{i+1}$  are aligned with  $\mathbf{n}_k$ .

### 4.6.4 Edge length regularization

To avoid short edges, we regularize selected length differences of adjacent edges. For example, if  $\mathbf{e}_i$  and  $\mathbf{e}_j$  are the edge vectors of two neighboring edges, we regulate the ratio of their lengths into a given interval, and  $E_{reg}^5$  is defined as:

$$\sum_{(i,j)} (\|\mathbf{e}_i\|^2 - r^2 \|\mathbf{e}_j\|^2 - \mu_{ij}^2)^2 + (\|\mathbf{e}_j\|^2 - r^2 \|\mathbf{e}_i\|^2 - \mu_{ji}^2)^2, \quad (4.10)$$

where the summation is over the pairs of edges chosen based on the used strip decomposition and  $r$  is set to 0.8 in our implementation. We can require a lower bound,

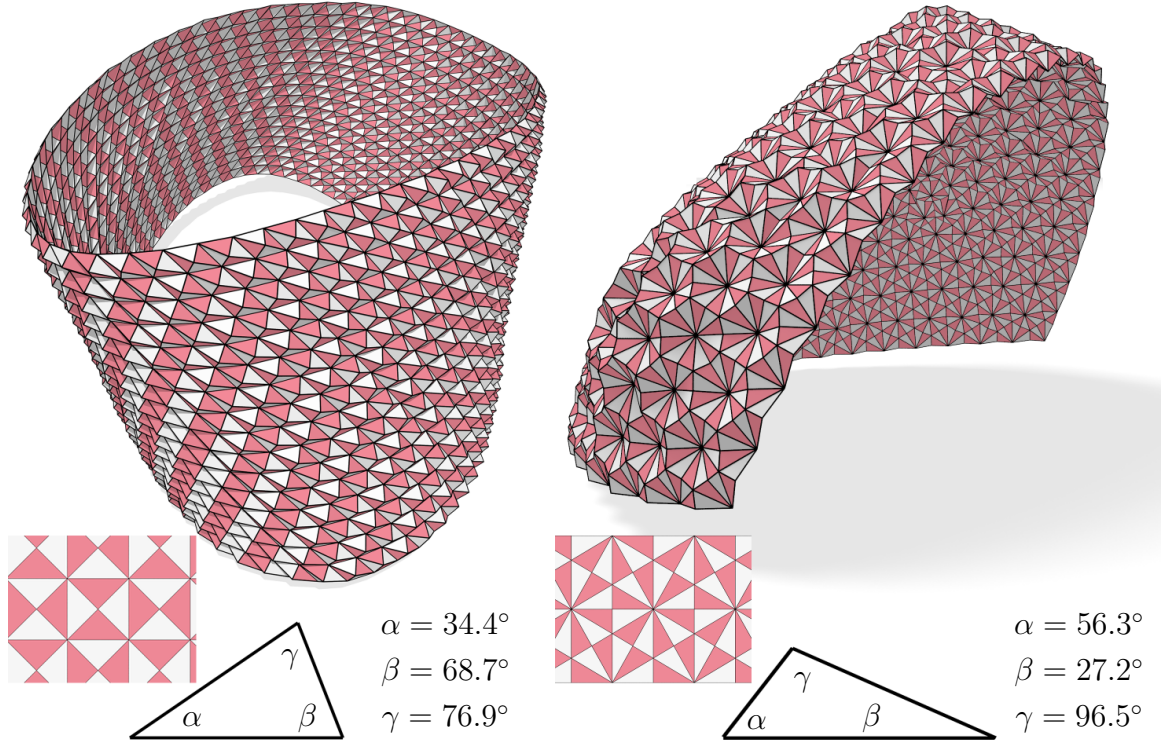


Figure 4.17: Triangle meshes constructed by a single type of triangle for the  $(4, 8, 8)^*$  pattern (left) and the  $(4, 6, 12)^*$  pattern (right) using symmetries with respect to an edge.

$l_{min}$ , on edge lengths by  $E_{reg}^6$ :

$$\sum_{i \in E} (\|e_i\|^2 - l_{min}^2 - \gamma_i^2)^2. \quad (4.11)$$

$\mu_{ij}$  and  $\gamma_i$  are slack variables for the inequality requirements.

## 4.7 Results

We present several results and discuss parameters, planarity, running time, failure cases, and comparison to related work.

**Models and patterns** We generate results for a set of selected surfaces showing multiple patterns per surface (see Figures 4.16, 4.22, and 4.23) where all patterns use the simplest strip decomposition. Throughout the chapter, we select the surfaces



Figure 4.18: We show a  $(3, 4, 6, 4)$  pattern on the Moomoo model with 57 singularities. This is the most complicated model shown in the chapter due to the high number of singularities (hexagons replaced by septagons and pentagons) and the high curvature variations.

to highlight the behavior of our regularizer in different situations. We select models to include regions of positive and negative Gaussian curvature, as well as interesting transition regions between them. We exemplify patterns on both open and closed surfaces, as well as surfaces with topological holes. We show a wood construction of an interior cladding of an architectural model in Figure 4.1. Finally, our method is used to generate rough patterns consisting of identical triangles (see Figure 4.17) and a  $(3, 4, 6, 4)$  pattern on a non-architectural model (see Figure 4.18). Note that our algorithm typically targets architectural models with moderate curvature variations and a sparse set of singularities. This model is therefore mainly depicted to demonstrate robustness.

**Strip decompositions** We demonstrate a pattern on a surface with different strip decompositions (see Figures 1.3 and 4.29). Different symmetries are used to accommodate each decomposition. Combining different strips and symmetries leads to a large variety of aesthetic results from a single basic pattern (see Figure 4.28).

**Planarity** All our models are successfully planarized. For assessment, we measure face planarity as the maximum distance between a vertex and a regression plane computed using PCA, normalized by dividing by the average edge length in the



model. Our tolerance for this measure is under  $10^{-2}$ . We illustrate one example in Figure 4.12 where we show how the Soumaya museum model is planarized. Before the optimization, many faces of the model are considerably non-planar. The optimization is nevertheless successful.

**Parameters** The main parameters stem from the configuration of the regularizer, i.e., the selected symmetries and strip decompositions. The weights for the different terms in the optimization vary slightly per pattern. In the optimization, we use the default parameters of 1.0 for face planarity, 0.1 for the closeness to the reference surface, and 0.01 for the symmetry-based regularizers. This choice is good for most examples. For example, the  $(3^4, 6)$  pattern in Figure 4.23, right and the  $(3, 4, 6, 4)$  pattern in Figure 4.16, left use these parameters without change.

Alternatively, the user can adjust the parameters to trade off regularity for stricter planarity or closeness to the reference surface; see Figure 4.19 for example. For the  $(4, 6, 12)$  patterns in Figure 4.22 and 4.23, we set the planarity to 5.0 and 10.0, respectively, while setting the closeness and symmetry parameters to the default ones. For the remaining terms, we advise that the edge-length parameter be equated with the symmetry regularization parameter, and the self-intersection avoidance parameter with the planarity parameter.

**Implementation details and running times** We implemented our framework in C++ using OpenMesh [48] for the mesh data structures and TAUCS [49] as the

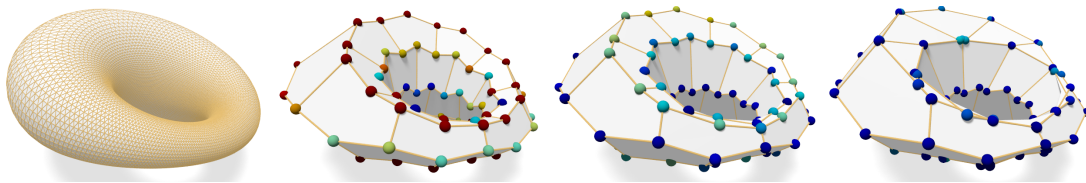


Figure 4.19: A planarized cyclide model. Hotter vertex colors indicate greater distances to the reference surface (left). A coarse input mesh can be successfully planarized and regularized at the cost of lower fidelity to the reference surface (second from left). Forcing closeness to the reference surface sacrifices regularity (second from right) and may cause edge degeneracies (right).

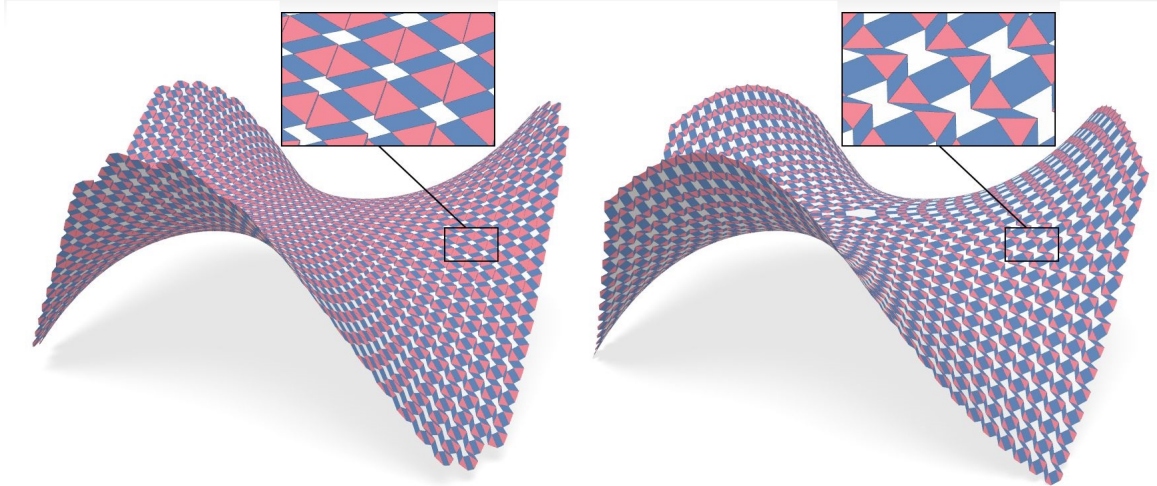


Figure 4.20: Failure case on a monkey saddle: due to the initialization used on the left, the pattern  $(3, 4, 6, 4)$  degenerates as some of the quads collapse to lines. With a different initialization that is better aligned with the principal curvature directions, the pattern can be mapped correctly (right).

library for sparse linear solvers. The running times for our examples are typically under one minute with an Intel Xeon X5550 2.67GHz processor. For example, the Soumaya model in Figure 4.12 has 7034 vertices and took 43 seconds to optimize. Smaller examples are much faster. For example, a  $(4, 6, 12)$  pattern on an HP surface with 432 vertices took 0.53 seconds to optimize.

**Failure cases and limitations** Our framework is sensitive to the triangle, quad or hex mesh that is used as input. A poor initialization leads to poor results. In Figure 4.20, we contrast the results of a poorly initialized optimization with a good initialization on the monkey saddle. In Figure 4.21, we show a poor initialization of the Soumaya model with many singularities in comparison with the example demonstrated above in Figure 4.12. In addition, we depend on the quality of existing code for the creation of the initial hex or quad patterns on the surface.

**Coarse meshes** Our algorithm can create adequate results on coarse meshes as well. However, we observe that coarse meshes can typically only be planarized and regularized when sacrificing the closeness to the reference surface, as demonstrated in Figure 4.19.

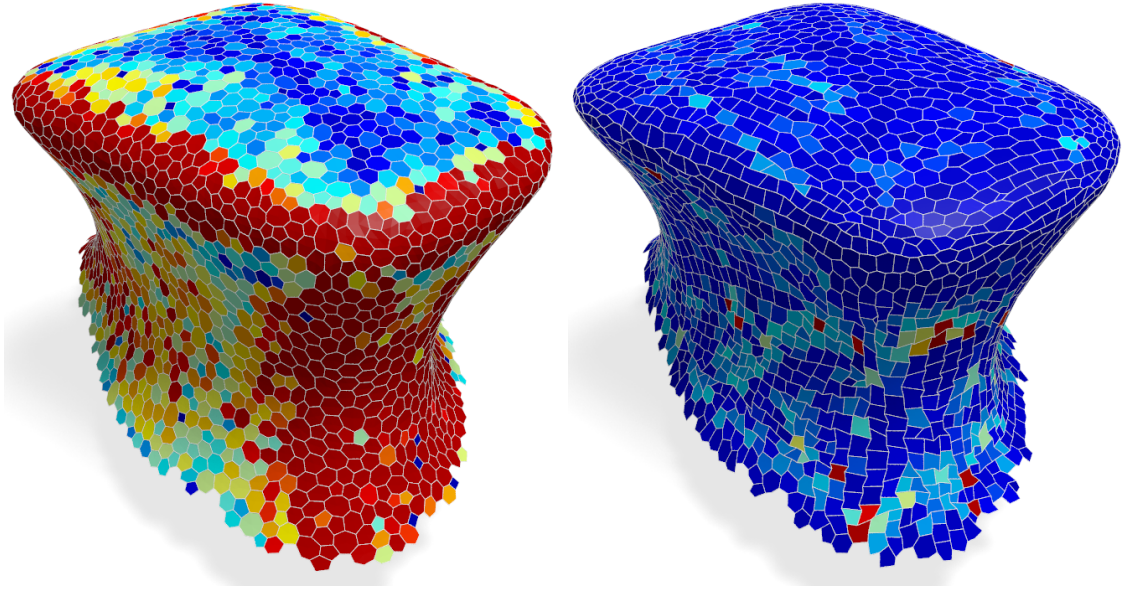


Figure 4.21: A failure case due to improper initialization. With a hex-dominant mesh dualized from of an arbitrary triangle mesh (left), the computed result (right) cannot achieve regularity or planarity, because there are too many singularities, leading to a over-fragmented strip decomposition.

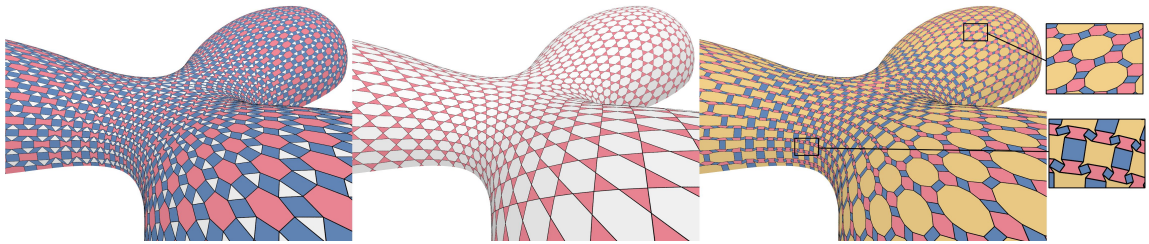


Figure 4.22: Left: A  $(3, 4, 6, 4)$  pattern using face symmetries. Middle: A  $(3, 6, 3, 6)$  pattern using vertex symmetries. Right: A  $(4, 6, 12)$  pattern using face symmetries.

**Comparison to PQ meshing** Our regularizers provide more degrees of freedom than does the traditional regularizer based on polyline fairness [29]. Therefore, our planarization is less sensitive to PQ meshes that are not initialized according to conjugate directions. We show examples in Figures 4.3, 4.26, and 4.28. A detailed analysis of the difference is provided in the additional materials.

**Comparison to PH meshing** Li et al. [40] proposed a regularizer for planar hex meshes, meshing the positively and negatively curved regions separately. However, they do not propose a specific solution for the transition region and cannot automat-

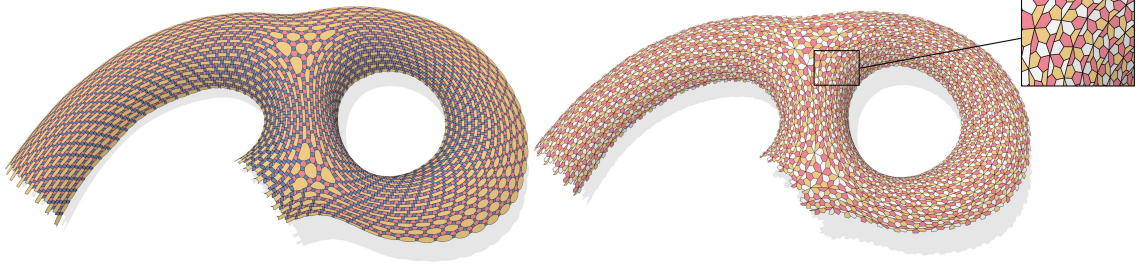


Figure 4.23: Semi-regular patterns on an architectural six shape. Left: A  $(4, 6, 12)$  pattern using face symmetries. A  $(3^4, 6)^*$  pattern using vertex symmetries. Note how prominent feature lines form automatically due to the regularization.

ically assign which regularizer to use. This may lead to artifacts in the transition region (see Figure 4.24) and to possible failures in the planarization (Figure 4.25). Our solution can produce significantly better results. It also caters to non-canonical strip decomposition. However, we note that Li et al. [40] propose a complete framework for PH remeshing, while our work focuses only on the regularizer used after the generation of an initial mesh layout.

**Comparison to ad-hoc regularizers** A large variety of regularizers has been proposed in other contexts. When using ad-hoc regularizers, typical problems occur, depending on how the regularizer is weighted. On the one hand, using a high weight leads to a mesh that is visually pleasing, but not polyhedral. On the other hand, using a low weight leads to a planar mesh that is highly irregular, including degeneracies like self-intersections. There is no effective weight that can achieve both planarity and regularity simultaneously. Figures 4.26 and 4.27 present examples.

## 4.8 Conclusions

We consider the design and optimization of polyhedral patterns, i.e. patterns of planar polygonal faces, on freeform surfaces. Our contributions are the description of a novel class of regularizers based on affine symmetries and a theoretical analysis of polyhedral patterns. In future work, we plan to study mixed patterns and their transition regions,

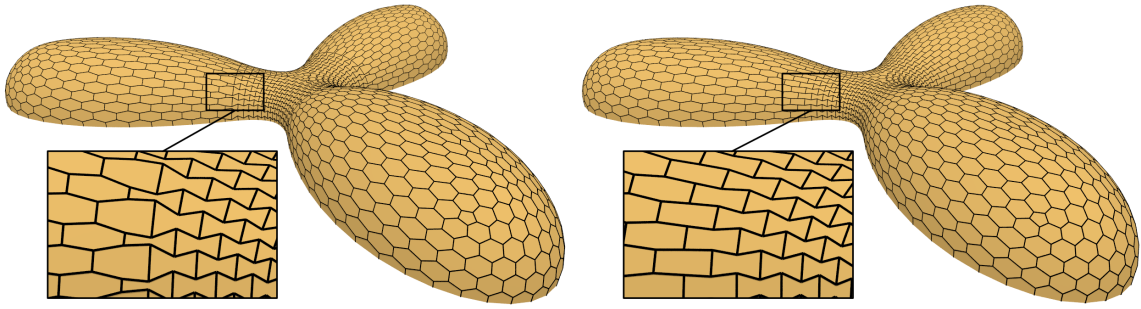


Figure 4.24: Comparison of Phex mesh aesthetics: with the same initialization, the regularizer of [40] generates the left mesh, while our approach leads to more natural transitions (right).

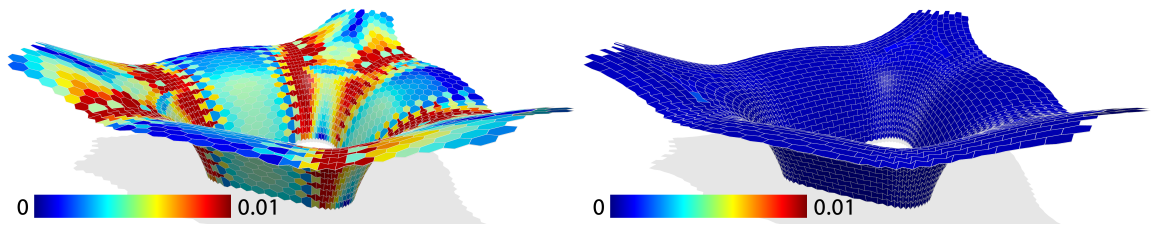


Figure 4.25: Comparison of Phex mesh planarity. Left: [40]. Right: our method achieves better planarity.

volumetric patterns such as frame structures for support in architectural applications, folding patterns, and time-varying polyhedral patterns for shading systems.

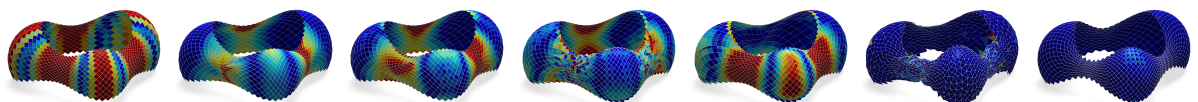


Figure 4.26: Mesh planarity. From left to right: initialization, polyline fairness, Laplacian, edge length, angles, face area, ours.

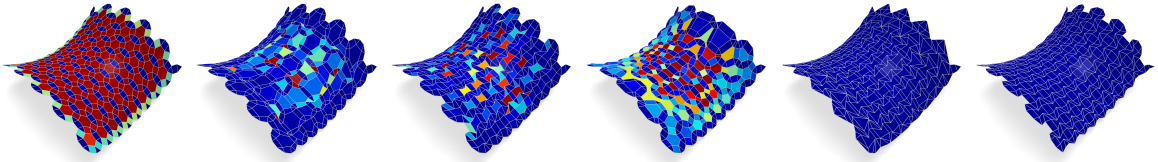


Figure 4.27: Mesh planarity. From left to right: initialization, Laplacian, edge length, angles, face area, ours.

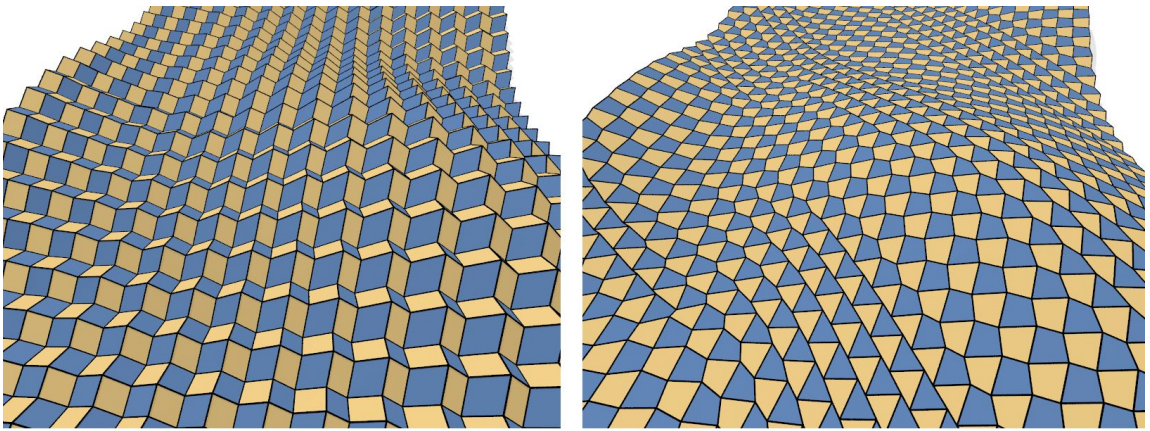


Figure 4.28: Different patterns generated by different symmetries. The pattern on the left uses face symmetries; the one on the right is based on edge midpoint symmetries.

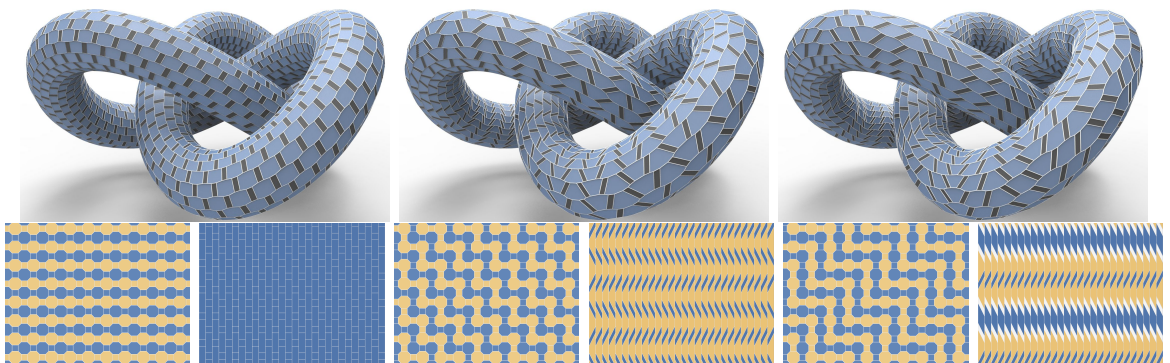


Figure 4.29: Three strip decompositions on the knot model for the  $(4, 8, 8)$  pattern. Figure 1.3 presents an explanation of the color coding.

# Chapter 5

## Space Frame Structures

### 5.1 Introduction

Space structures, also called space frames or space frame structures, are elegant and materially efficient *truss*-like structures consisting of beams (*two-force members*) connected at nodes. Working with space structures is desirable and often necessary in industrial and architectural constructions. The design and optimization of space structures present many challenges, especially for designs with complex geometry.

In industrial design, space structures have been widely used for structures that should be light weight but also statically sound, such as bikes, cars, or airplanes. A major advantage of space structures is their static soundness with a small amount of material usage. In many situations, space structures also enable a simple manufacturing process by assembling or welding beams or bars. Moreover, the richness of the design space of space structures allows for the possibility of finding elegant structures to support a variety of different designs with highly customized shapes. More recently, similar ideas have also been extended to 3D printing and personalized design and fabrication.

In architectural construction, space structures often serve as statically sound supportive structures approximating target shapes or underlying desired freeform surfaces. In many prominent projects, they are not visible. However, they are essen-

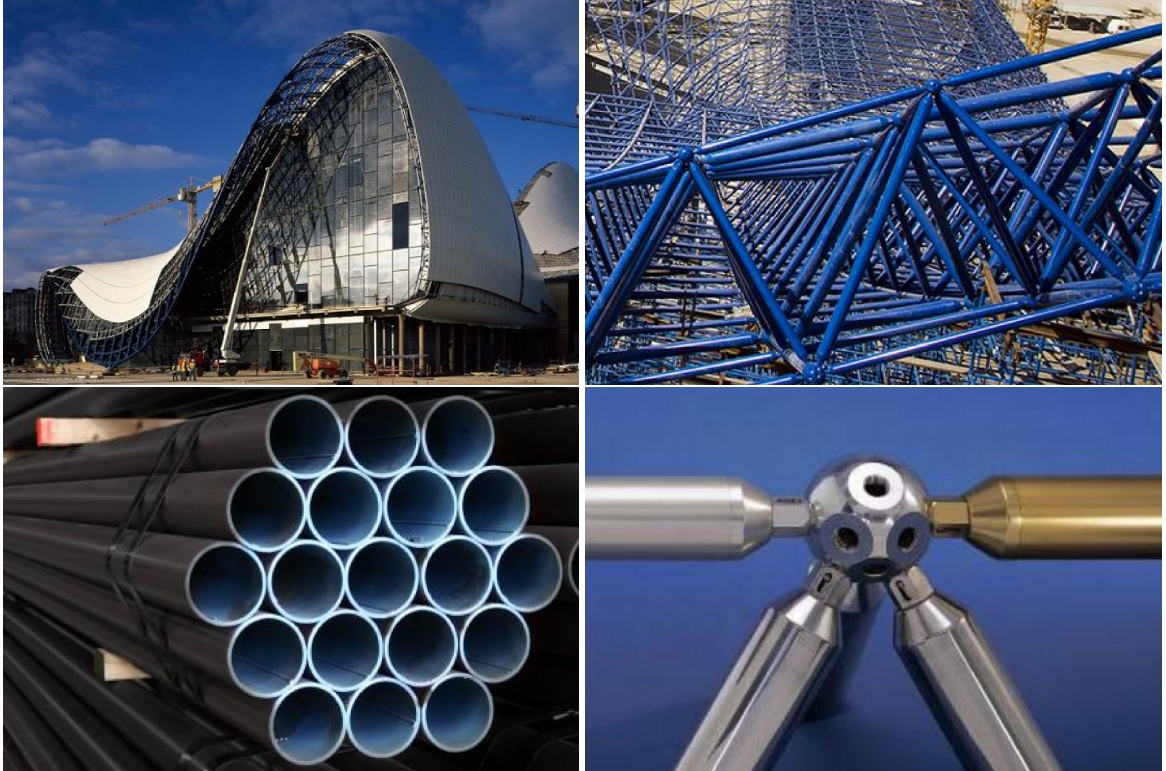


Figure 5.1: The Heydar Aliyev Cultural Center (top-left) in Baku, Azerbaijan is designed by Zaha Hadid. A supporting space structure (top-right) underlying the freeform surface is constructed by MERO-TSK using circular hollow section tubular beams (bottom-left) and the KK-Ball Node System (bottom-right).

tial for the realization of the architectural design. For example, the Heydar Aliyev Cultural Center in Azerbaijan designed by Zaha Hadid, is constructed based on an underlying space structure, as shown in Figure 5.1.

Besides serving as underlying supportive structures, many space structures are also visible. For these visible space structures, aesthetics considerations, such as simplicity and regularity, should also be considered. These space structures are not limited to freeform architectural designs like stadiums or cultural centers, but also commonly seen on bridges, towers, and even playground domes.

Despite the universal applicability of space structures, most of the current conventional design processes are based on manually exploring different variations using interactive editing and customized scripting. There are usually many iterations of



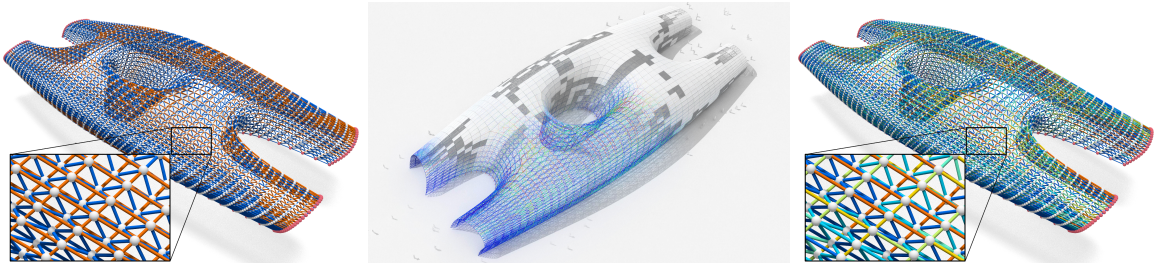


Figure 5.2: Two different construction solutions of a statically sound space structure approximating the train station model. The solution on the left uses two types of customized tubular beams, and the solution in the middle and the right uses six types. Similar to Figure ??, the beams are color coded by their cross-section types, where hotter colors represent stronger beams. For aesthetics, these beams have the same outer radius, as the cross-section areas are controllable by the thicknesses of the circular hollow sections (shown in Figure 5.1, bottom-left).

design and verification to optimize the connectivities, node positions, and the cross sections of beams. Based on our collaboration with structural engineers working on architectural projects in industry, we narrowed down the challenges in the design of space structures to the following aspects. (Goal 1) First, the structure should be statically sound. This means that the structure is in force equilibrium with axial forces along the beams without bending moments. (Goal 2) Second, the structure should be aesthetically pleasing, constructed with regularly arranged beams and nodes. (Goal 3) Third, the structure should approximate a given designed shape, e.g., a freeform architectural surface. (Goal 4) Fourth, the cost of the structure should be minimized. The most important factor associated with cost is the material usage. Therefore, a cost-effective space structure consists of beams with variable cross sections, since most material is used for the beams. Further, beams are usually manufactured by extrusion (aluminium) or bending and welding (steel), followed by cutting, so that it is important that many beams share the same cross section.

We propose a systematic framework to design and optimize statically sound space structures that approximate freeform surfaces. To achieve the desired goals mentioned above, our framework allows a user to explore different connectivities of the space

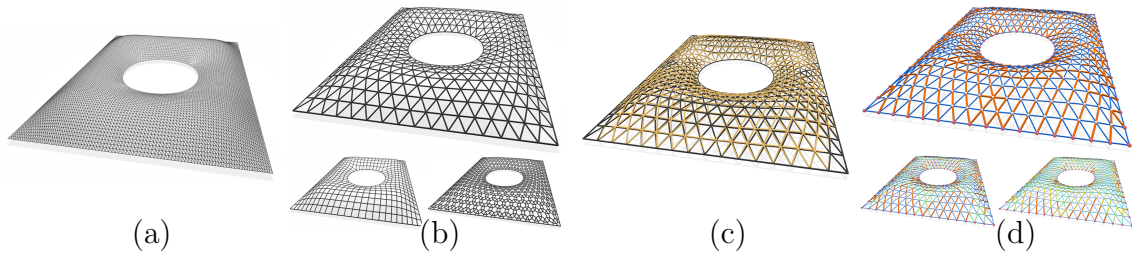


Figure 5.3: Framework overview: a) Based on an input reference mesh, our system provides a set of tools for creating initial structures with different connectivities. b) Three possible connectivities modeled with our system. c) After initialization, we optimize the closeness to the reference shape, boundary alignment, static soundness, total volume consumption, and geometric regularity, by adjusting vertex locations and axial forces. d) We further adjust the beam types to minimize the total volume of material used under the condition that the structure should be constructed with beams of a limited number of cross section types that could be customized. Here, we show solutions for 2, 3, and 6 beam types.

structures, optimize the node positions, and adjust the beam cross sections. Our contributions to the design and optimization of space structures include:

- A novel system for the design and optimization of space structures that considers the most important practical aspects.
- A break down of the overall problem into subproblems formulated into manageable optimization problems.
- A practical optimization algorithm that efficiently tackles a challenging mixed-integer programming problem with a bilinear objective function and quadratic constraints.

## 5.2 Previous Work

Studies related to our work can be found in different disciplines. Design and optimization of trusses with relatively simple geometries have been studied for a long time in structural engineering. Recently, the design of statically sound structures at different

scales has been studied in the context of 3D printing and architectural geometry in the graphics community.

More than one century ago, Anthony Michell, in his seminal article [50], provided sufficient conditions and analytical examples for special types of optimal trusses to attain their maximal efficiency in terms of material usage. Michell's work was based on ideal assumptions that all external forces are determined and imposed on discrete vertices and that beams are infinitesimally thin elements.

The analytical results of Michell were later reproduced by many computational studies on topological optimization. The most commonly used algorithm for topology optimization of trusses is the *ground structure method* [51,52]. It starts from a densely connected structure with fixed node positions and given applied forces. To reduce the total amount of material consumption, it adjusts the beam cross-section areas to remove beams and simplify the graph. However, the ground structure method typically generates connectivities that are infeasible for construction, which renders manual adjustment of the final connectivity necessary. Many references in this field have been extensively reviewed in the thesis of Mitchell [53].

Recently, researchers in the graphics community started to look into the design of statically sound structures. For the purpose of procedural modeling to create structurally realistic models in a virtual world, Smith et al. [54] studied the design of truss-like structures. Similar goals were also studied on a smaller scale in the context of 3D printing. For example, Wang et al. [55] studied the reduction of material usage through frame structures. After *Thrust Network Analysis* was introduced in the thesis of Block [56], there have been many studies on the design of freeform self-supporting surfaces. Among them, the thrust networks in [57–59] are auxiliary and invisible in the final constructions, without aesthetic restrictions (Goal 2). Vouga et al. [60] and Tang et al. [29] studied thrust networks that are visible in the final construction, but they did not consider the reduction of material consumption by adjusting the

cross-section areas of beams (Goal 4). It is worth mentioning that compared with our work, these works are subject to the additional restriction that only compressive forces are allowed.

Besides static soundness and the reduction of material usage, complexity of the construction process should also be considered in the design phase. To simplify the construction of a large structure, it is often necessary to enforce the rule that the structures are constructed by repetitive elements so that they can be batch manufactured using the same factory settings. Therefore, the work of Fu et al. [61] aimed to enable the paneling of a freeform surface using limited types of faces. Jiang et al. [42, 62] studied *freeform honeycomb structures* with repetitive nodes and *Lobel frames* with edges of the same length. Similar goals are also pursued in [23] and [46]. However, using beams of repetitive lengths aggressively diminishes the shape space, and this is often unnecessary for space structures where the beams can be easily cut after extrusion or bending and welding. Therefore, our aim for repetitive cross-section areas is more relevant to the reduction of cost, while imposing no restriction on either statics or aesthetics.

### 5.3 Overview

We provide users a framework to create space structures, optimize for static soundness, and minimize the total volume of material used. The framework is comprised of four stages, including connectivity enumeration, force initialization, optimization of node positions, and discrete optimization of beam cross sections, as illustrated in Figure 5.3.

**Connectivity Enumeration, Ranking, and Editing** The input to our system includes a reference model given as a finely tessellated triangle mesh and desired boundary curves. We provide a set of tools to generate the connectivity of the space

structure:

- Coarse mesh enumeration, ranking, and editing.
- Subdivision for mesh refinement and pattern generation.
- Construction of multi-layer structures.

**Force Initialization** Once connectivity and node positions are initialized, we estimate the axial force densities of beams, defined as forces per unit length. Space structure design should minimize the bending moments of beams such that static soundness of the structure should be achieved with only axial forces, within the bounds determined by beam types and cross-section areas. Therefore, we solve a constrained least squares problem to achieve force balance in a least-squares sense with axial forces only. The estimated axial force densities are the input for the next stage of the optimization, where force equilibrium is strictly required.

**Optimization of Node Positions** The next stage of computation is a multi-objective, nonlinear optimization in terms of node positions. Under constraints encoding the material properties of the beams and safety factors of buckling control, the computation relocates the node positions while adjusting the axial forces accordingly for better performance. A user can specify hard constraints that the computation has to strictly satisfy, such as keeping selected nodes fixed, or soft constraints, such as closeness to the reference surface, fairness of the structure, and a relaxed formulation of total material usage.

**Discrete Optimization of Beam Cross Sections** While constructing space structures, we would like to minimize the total material consumption by adjusting the beam cross-section areas. At the same time, space structures should be constructed with beams of limited types of repetitive cross-section areas, with adjustable beam cross-section areas of each type. This problem involves discrete variables (assignments of types), and continuous variables (beam cross-section areas). With an optimization

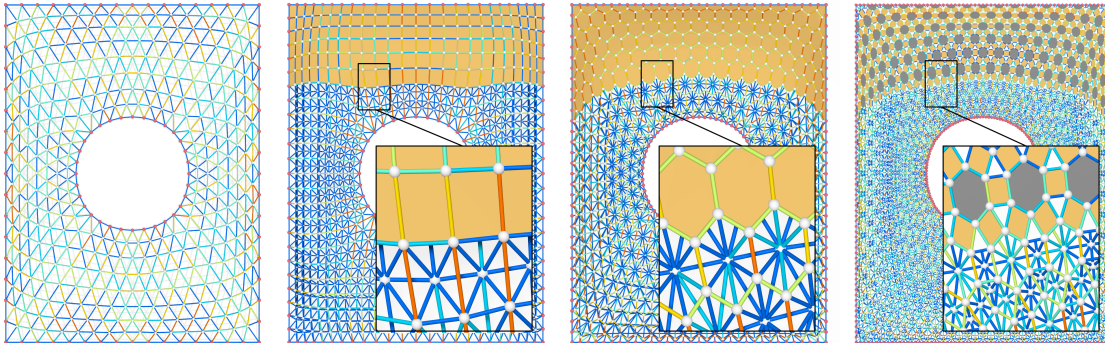


Figure 5.4: To enrich the design space, our framework provides a user with a set of tools to create base meshes with a variety of connectivities. Here, we show space structures approximating the British Museum model with a base mesh being a triangle mesh (left), a quadrilateral mesh (second from left), a hexagonal mesh (second from right), and a semi-regular pattern consisting of triangles, quadrilaterals, and hexagons (right). Each of the structures is constructed with tubular beams with six types of customized cross sections.

scheme alternating with continuous and discrete variables, we can successfully tackle this problem, overcoming the restrictions of general-purpose, mixed-integer programming solvers.

## 5.4 Optimization Framework

The inputs to our framework include a reference surface given as a fine triangle mesh,  $M$ , and boundary curves represented as polylines or splines,  $C^i, i = 1 \dots n^C$ . Our goal is to design and optimize a space structure,  $S$ , which consists of a set of nodes (vertices),  $V$ , connected by a set of beams (edges),  $E$ . The supported nodes are  $V^f \subset V$ , and the boundary vertices are  $V^B \subset V$ , which may also be supported. We denote  $\bar{V} \subset V$  as the nodes of the outer layer which should approximate the reference surface. In single-layer structures,  $\bar{V} = V$ .

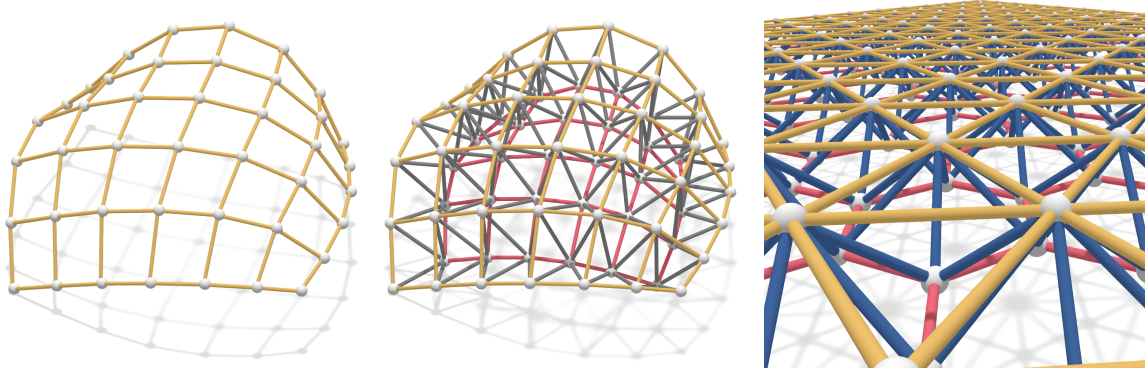


Figure 5.5: Left and middle: a double-layer quadrilateral space structure is constructed based on a quadrilateral base mesh. Right: a double-layer space structure based on a triangle mesh.

#### 5.4.1 Connectivity Enumeration, Ranking, and Editing

As connectivity is an essential element in space-structure design, we provide a set of connectivity modeling tools. This set of tools includes connectivity enumeration, interactive editing, and subdivision.

**Coarse-mesh enumeration** Coarse-mesh enumeration can be used for the outer layer. For space structures with quadrilateral base meshes, the connectivities of the base layer meshes are generated based on [63]. Triangle meshes are created by CVT-based remeshing [64].

**Interactive editing** Interactive editing allows the user to sketch a desired coarse mesh connectivity. A user can add, remove, and relocate vertices and edges. A user also can edit a mesh by adding or removing polylines and relocating singularities.

**Connectivity refinement** A user can select from a set of subdivision and procedural rules to refine the mesh. Simple subdivision rules include Loop and Catmull-Clark. Procedural refinement rules include the construction of hexagonal meshes and semi-regular patterns [65]. In multi-layer structures, derivation rules are applied based on single-layer meshes [66] as shown in Figure 5.5.

Figure 5.4 illustrates different connectivities created using a combination of the methods described above for the model of the British Museum. All of the space

structures are optimized using the later stages of our framework and are structurally sound.

### 5.4.2 Force Initialization

A space structure must be statically sound. In the following, we first describe our structural assumptions and then introduce how axial forces are initialized.

**Structural Assumptions** In space structure design, we aim for structures in static equilibrium with minimal bending moments of the beams. Therefore, axial forces alone should be in force balance with the loads. Similar to previous work, e.g., [60], the dead loads of the nodes on the outer layer are assumed to be proportional to the influence areas, i.e., the areas of the dual cells, of the shell supported by the space structure, as shown in Figure 5.6. Here, we also assume that the weights of the beams are much less than the weights of the panels that the structure has to support. Alternative assumptions do not essentially change the workflow.

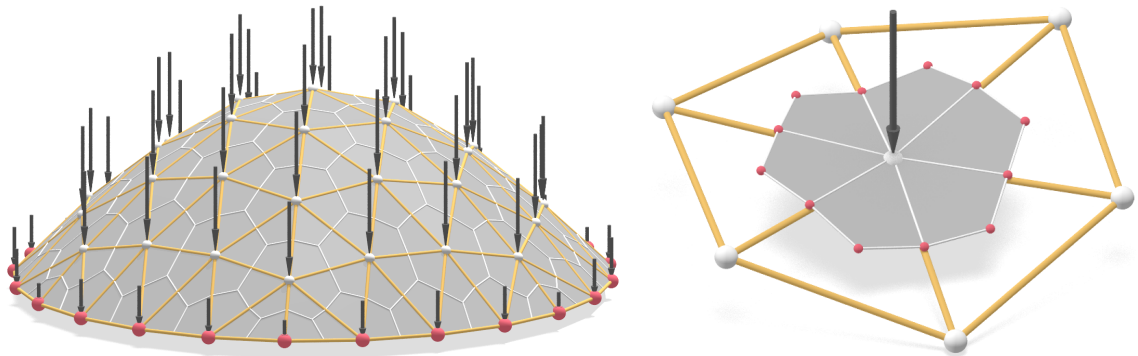


Figure 5.6: A space structure (left) supporting a shell like surface is in force balance under appropriate structural assumptions, e.g., the load at every node is proportional to the area of its dual cell (right). Red dots in the left indicate supported nodes.

**Force Initialization** To proceed to the next stages of computation, all the variables need to be properly initialized. The missing variables are the axial force densities. The force densities,  $w_{ij}$ , are the axial forces per unit length defined on each beam. The axial forces at each node,  $\mathbf{v}_i$ , should balance the load,  $\mathbf{l}_i$ , imposed on the node. These



forces are precomputed according to the structural assumptions described above. This is equivalent to the minimization of the energy term encoding force equilibrium,

$$\sum_{j: \{i,j\} \in E} w_{ij}(\mathbf{v}_j - \mathbf{v}_i) = -\mathbf{l}_i, i = 1, \dots, |V|. \quad (5.1)$$

If there is no solution, the system is solved in a least-squares sense. If there are multiple solutions, a least-norm solution is computed. In the next stage of computation, the node locations are treated as variables optimized together with the axial forces.

### 5.4.3 Optimization of Node Positions

After the connectivity of a space structure has been determined, the locations of the nodes should be further adjusted together with the axial forces. We model an objective function consisting of five terms: closeness to the reference surface ( $E_{close}$ ), boundary alignment ( $E_{boundary}$ ), static equilibrium ( $E_{static}$ ), total volume ( $E_{volume}$ ), and geometric regularity ( $E_{reg}$ ).

**Closeness to Reference Surfaces** The outer layer of the space structures is required to approximate the given reference shape provided as the fine triangle mesh,  $M$ . The closeness of a vertex,  $\mathbf{v}_i \in \bar{V}$ , to the reference surface,  $M$ , is required by constraining  $\mathbf{v}_i$  to move only on the tangent plane associated with its closest point,  $\mathbf{v}_i^*$ , on the reference mesh,  $M$ :

$$E_{close} = \sum_{\mathbf{v}_i \in \bar{V}} ((\mathbf{v}_i - \mathbf{v}_i^*) \cdot \mathbf{n}_i^*)^2. \quad (5.2)$$

Here, the vector,  $\mathbf{n}_i^*$ , is the unit normal vector of the tangent plane. Figure 5.7 illustrates the effects of the closeness term.

**Boundary Alignment** We also require that the boundary vertices stay on the de-

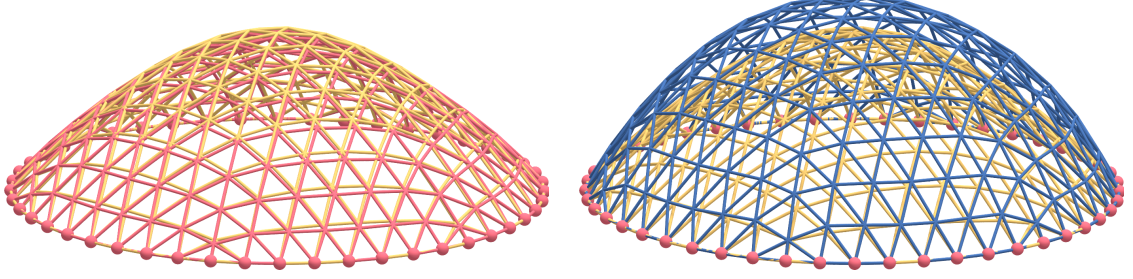


Figure 5.7: Optimization of a single-layer space structure with the energy term enforcing closeness (left) provides a better approximation to the reference surface colored in yellow. In contrast, optimization without the closeness term (right) drives the structure away from the initial design to further reduce the total volume of material used.

sired boundary curves,  $C^i$ , by enforcing that a boundary vertex,  $\mathbf{v}_i \in V^B$ , can only move along the tangent line at its projected point,  $\mathbf{v}_i^\dagger$ , on the reference curve:

$$E_{boundary} = \sum_{\substack{\mathbf{v}_i \in V^B \\ j \in \{1,2\}}} \left( (\mathbf{v}_i - \mathbf{v}_i^\dagger) \cdot \mathbf{n}_{i,j}^\dagger \right)^2. \quad (5.3)$$

Here,  $\mathbf{n}_{i,1}^\dagger$  and  $\mathbf{n}_{i,2}^\dagger$  are unit vectors that are orthogonal to each other and to the tangent vector at the projected point,  $\mathbf{v}_i^\dagger$ .

**Static Equilibrium** The axial forces are related to both the total volume of the material consumption and force balance. Absolute values of the force density terms,  $|w_{ij}|$ , are needed to represent the total volume. Therefore, we introduce slack variables,  $u_{ij}^+$  and  $u_{ij}^-$ , where  $(u_{ij}^+)^2 = \max(0, w_{ij})$  and  $(u_{ij}^-)^2 = \max(0, -w_{ij})$ . With these new variables,  $w_{ij} = (u_{ij}^+)^2 - (u_{ij}^-)^2$  and  $|w_{ij}| = (u_{ij}^+)^2 + (u_{ij}^-)^2$ . The energy term for static equilibrium,  $E_{static}$ , is:

$$\sum_{i \in V} \left( \sum_{j: \{i,j\} \in E} \left( (u_{ij}^+)^2 - (u_{ij}^-)^2 \right) (\mathbf{v}_j - \mathbf{v}_i) + \mathbf{l}_i \right)^2. \quad (5.4)$$

**Total Volume** The minimal beam cross-section areas are linearly related to the absolute values of axial forces. Therefore, each solution of axial forces corresponds

to a different assignment of beam cross-section areas. With the introduced slack variables, the total volume of material used can be written as:

$$E_{volume} = \sum_{i,j: \{i,j\} \in E} ((u_{ij}^+)^2 + (u_{ij}^-)^2) \|\mathbf{v}_j - \mathbf{v}_i\|^2. \quad (5.5)$$

**Geometric Regularizers** For aesthetics, a space structure should also be regular. For a space structure with a triangle mesh as its base mesh, a uniformly weighted Laplacian term is enforced on each layer for smoothness. For a structure with a quadrilateral mesh as its base, polyline fairness energy is applied. For a space structure constructed based on a semi-regular pattern, symmetry-based regularizers proposed in [65] are used.

**Further Considerations** Additionally, the following objectives, which are currently not implemented in our system, could easily be included:

- Alignment with respect to stress fields.
- Maintenance of layer offsets for multilayer structures.
- Fairness of axial forces for reduced stress concentration.
- Coplanarity of selected vertices.

**Optimization Algorithm** The sum of the energy terms is solved by a quasi-Newton method [67], based on an implementation of the Hybrid Limited-memory Broyden-Fletcher-Goldfarb-Shanno (HLBFGS) method developed in [64]. For the sum of energy terms,  $E_{sum} = \sum_i \lambda_i E_i$ , we choose higher weights, 1, for closeness and force equilibrium, and lower weights, 0.01, for the other terms. At this stage, we have not considered the practical construction complexity that the beams should have repetitive cross-section areas. Therefore, the next stages of computation based on discrete optimization are necessary for material minimization.

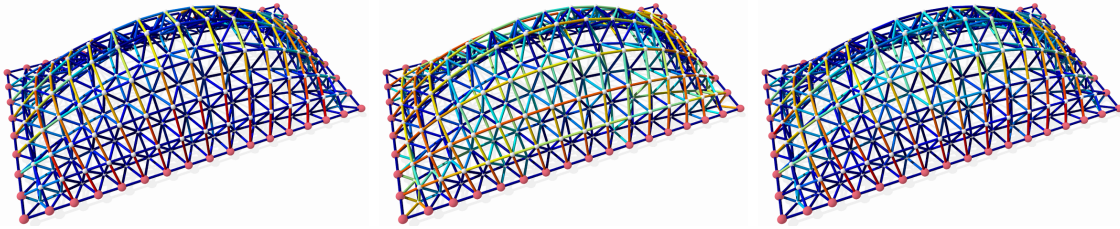


Figure 5.8: Three different assignments of axial forces achieving static equilibrium for a statically indeterminate (hyperstatic) structure with fixed nodes and given loads. The beams are color coded according to the absolute values of the axial forces, where red indicates stronger beams. Typically, there is an infinite space of such valid solutions that keep a space structure in static equilibrium.

#### 5.4.4 Discrete Optimization of Beam Cross Sections

The solution obtained from the previous stage assumes that the cross-section areas of the beams can be arbitrarily and continuously adjusted. While the total material consumption is reduced, such a solution imposes a high cost for beam customization, as every beam has to be manufactured with a different factory setting. However, a lot of material would be wasted if all beams had the same cross section. Therefore, as a compromise, we create structures based on a small number, e.g., 2, 3, or 6, of types of cross sections.

To keep the problem manageable, the previous nonlinear formulation needs to be simplified before introducing discrete variables. To reduce the complexity, we fix the node positions and solely focus on the axial forces and beam cross-section areas. Such simplification is feasible as the structures we study are *statically indeterminate* (or *hyperstatic*). This means that the solution space of axial forces in force equilibrium generally has a nontrivial dimension for fixed nodes and given loads as illustrated in Figure 5.8. In the infinite space of valid solutions, there is typically one that has minimal volume for beams of varying cross sections. However, this solution is not necessarily optimal when beams are chosen from a fixed set of cross sections. Therefore, we cannot simply solve a continuous problem and take the solution as a

basis for the discrete problem.

The natural formulation of the optimization is a mixed integer programming problem. However, there is no generic method to solve such a problem. In the following, we first present the mixed integer programming problem. We then discuss our novel solution to tackle this problem in a practical manner.

### Mixed Integer Programming Formulation

$$\underset{x_{ij}, \bar{a}_j, s_i}{\text{minimize}} \quad \sum_i l_i \sum_{j=1}^k \bar{a}_j x_{ij} \quad (5.6)$$

$$\text{subject to} \quad B^T \mathbf{s} = -\mathbf{f} \quad (5.6a)$$

$$\sum_{j=1}^k x_{ij} \bar{a}_j + s_i \geq 0; \quad i = 1, \dots, |E| \quad (5.6b)$$

$$\sum_{j=1}^k x_{ij} \bar{a}_j - s_i \geq 0; \quad i = 1, \dots, |E| \quad (5.6c)$$

$$\sum_{j=1}^k x_{ij} \leq 1; \quad i = 1, \dots, |E| \quad (5.6d)$$

$$x_{ij} \in \{0, 1\}; \quad j = 1, \dots, k, \quad i = 1, \dots, |E|. \quad (5.6e)$$

Here, the coefficients,  $l_i$ , are the lengths of the beams. Here, the scalars,  $\bar{a}_j, j = 1, \dots, k$ , are the bounds of the axial forces of the  $k$ -beam types. The assignment of beams types is reflected by the integer variable,  $x_{ij}$ , which is 1 if the  $i$ -th beam is assigned with the  $j$ -th type, and 0 otherwise. Each element of  $\mathbf{s} \in \mathbb{R}^{|E|}$  is a signed scalar value of the axial force of a beam, proportional to the beam length and the force density:  $s_i = w_i l_i$ .

The linear constraints,  $B^T \mathbf{s} = -\mathbf{f}$ , for force balance on axial forces is equivalent to Equation 5.1 for force densities. The matrix,  $B^T \in \mathbb{R}^{3|V| \times |E|}$ , is called the *nodal equilibrium matrix*. It represents the connectivity of the space structure, converting

axial forces on beams to the resultant forces. The vector,  $\mathbf{f} \in \mathbb{R}^{3|V|}$ , represents all the loads,  $\mathbf{l}_i, i = 1, \dots, |V|$ .

**Difficulties of Mixed Integer Programming** The seemingly simple formulation presented above turns out to be remarkably difficult to solve with a generic solver for mixed integer programming. This is due to the fact that both the objective function and the constraints are quadratic. Besides, the objective function is generally not positive definite. With discrete variables, the problem becomes even more challenging.

There are three sets of variables: the axial forces of beams,  $s_i$ , the cross-section areas of beam types  $\bar{a}_j$ , and the assignment of types for beams  $x_{ij}$ . We propose to break the overall problem into individual subproblems and solve them in an alternating scheme. Studying the structure of the problem, we derive the following three subproblems:

- Sp-1: Fix  $\bar{a}_j$ , and solve for  $s_j$  and  $x_{ij}$ .
- Sp-2: Fix  $s_j$ , and solve for  $\bar{a}_j$  and  $x_{ij}$ .
- Sp-3: Fix  $x_{ij}$ , and solve for  $s_j$  and  $\bar{a}_j$ .

In the following, we discuss how these three subproblems are formulated and solved. We then discuss the overall algorithm comprised of these three subproblems.

**Sp-1** (fix  $\bar{a}_j$ ): When the cross-section areas of the  $k$ -beam types,  $\bar{a}_j$ , are given, Equation 5.6 is reduced to a standard mixed integer *linear* programming problem in terms of  $\mathbf{s}$  and  $x_{ij}$ . However, directly applying a generic solver has a very poor scalability, and it soon becomes impractical even for a problem of moderate scale, as shown in Section 5.5.

*Linear programming relaxation.* To further reduce the complexity, we relax the integer variables,  $x_{ij}$ , to real variables [68]. Instead of requiring  $x_{ij}$  to be either 0 or

1, we allow them to be continuously adjustable between 0 and 1:

$$\begin{aligned}
& \underset{x_{ij}, s_i}{\text{minimize}} && \sum_i l_i \sum_{j=1}^k \bar{a}_j x_{ij} \\
& \text{subject to} && (5.6a) - (5.6d) \\
& && x_{ij} \geq 0.
\end{aligned} \tag{5.7}$$

In Equation 5.7, each sum,  $\sum_{j=1}^k \bar{a}_j x_{ij}$ , can assume values between 0 and  $\max(\bar{a}_j, j = 1, \dots, k) := a_{max}$  with non-unique choices of  $x_{ij}$ . Therefore, we further eliminate  $x_{ij}$  by introducing  $a_i = \sum_{j=1}^k \bar{a}_j x_{ij}$  and further simplify the problem to the following continuous linear programming problem:

$$\underset{a_i, s_i}{\text{minimize}} \quad \sum_{i=1}^{|E|} l_i a_i \tag{5.8}$$

$$\text{subject to} \quad B^T \mathbf{s} = -\mathbf{f} \tag{5.8a}$$

$$a_i + s_i \geq 0; \quad i = 1, \dots, |E| \tag{5.8b}$$

$$a_i - s_i \geq 0; \quad i = 1, \dots, |E| \tag{5.8c}$$

$$a_i \leq a_{max}; \quad i = 1, \dots, |E|. \tag{5.8d}$$

**Sp-2** (fix  $s_j$ ): When axial forces are given, we seek the cross-section areas of the beam types,  $\bar{a}_j$ , and beam-type assignments,  $x_{ij}$ . We sort the beams according to the absolute values of axial forces,  $|s_i|, i \leq |E|$ , in a nondecreasing order and relabel them accordingly. Next, we look for optimal cuts at the indices,  $m_t \in \mathbb{Z}^+, t = 1 \dots k - 1$ , so that the volume is minimized:

$$\underset{m_t \in \mathbb{Z}^+, t=1 \dots k-1}{\text{minimize}} \quad \sum_{i=1}^k \left( \sum_{j=m_{i-1}+1}^{m_i} l_j |s_{m_i}| \right). \tag{5.9}$$

By definition, the boundaries of the cuts are  $m_0 = 0$ , and  $m_k = |E|$ . The internal cuts,  $m_i, i = 1 \dots k - 1$ , are initialized equidistantly between  $m_0$  and  $m_k$ , as shown

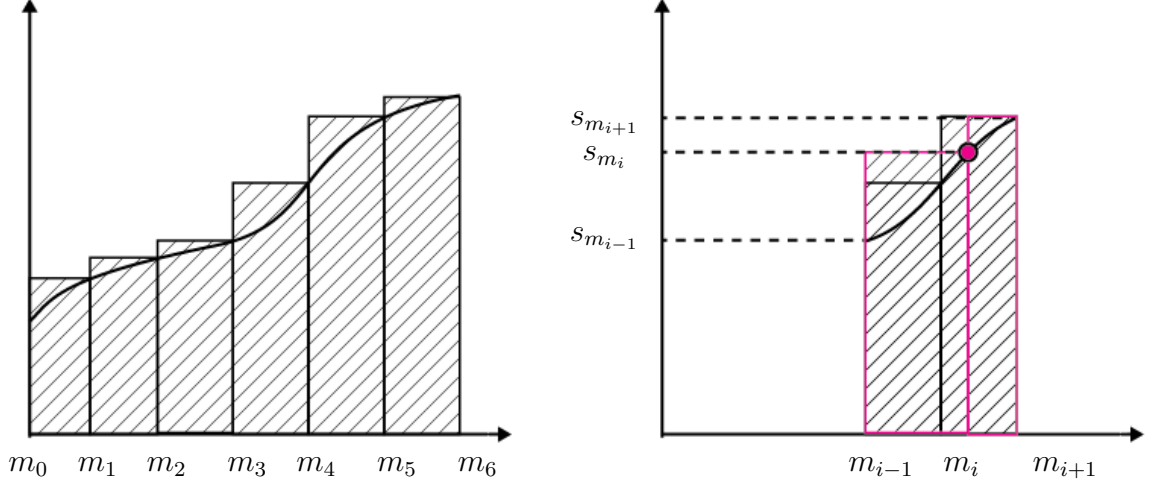


Figure 5.9: Left: In Sp-2, we sort the beams according to the absolute values of axial forces,  $|s_i|, i \leq |E|$ , in a nondecreasing order and seek the optimal cuts to determine the beam types. Right: We start from uniformly sampled cuts and adjust each cut between its neighbors while fixing the other cuts iteratively.

on the left side of Figure 5.9. Next, each individual cut is adjusted by sweeping in the range bounded by its two neighboring cuts, while fixing the other cuts, as shown on the right side of Figure 5.9. This routine of seeking the optimal cuts with given balanced forces generally converges in fewer than 10 iterations.

**Sp-3** (fix  $x_{ij}$ ): Next, we consider the case when the assignment of beams is fixed in Equation 5.6. For better clarity, we denote  $\Phi_j := \{i | x_{ij} = 1\}$  as the set of beams that is assigned with the  $j$ -th type:

$$\begin{aligned}
 & \underset{s_i, \bar{a}_j}{\text{minimize}} && \sum_{j=1}^k \left( \sum_{i \in \Phi_j} l_i \right) \bar{a}_j \\
 & \text{subject to} && B^T \mathbf{s} = -\mathbf{f} \\
 & && \bar{a}_1 + s_i \geq 0, \bar{a}_1 - s_i \geq 0; \quad i \in \Phi_1 \\
 & && \dots \\
 & && \bar{a}_k + s_i \geq 0, \bar{a}_k - s_i \geq 0; \quad i \in \Phi_k
 \end{aligned} \tag{5.10}$$

As coefficients of  $\bar{a}_j$ , the sums,  $\sum_{i \in \Phi_j} l_i$ , are constants. This problem is thus a standard linear programming problem.



**Overall Algorithm** The three subproblems are assembled together for an overall practical algorithm, constituting three stages. First, as a preprocessing step, we determine the bounds of  $a_{max}$ , which could be used as input for Sp-1 (fix  $\bar{a}_j$ ). Next, as the main procedure, the algorithm alternates between Sp-1 (fix  $\bar{a}_j$ ) and Sp-2 (fix  $s_j$ ) to find the optimal beam type assignment,  $x_{ij}$ . Finally, as a post-processing step, we use Sp-3 (fix  $x_{ij}$ ) to further adjust the cross-section areas of each type,  $\bar{a}_j$ .

*Pre-processing:* To find the range of feasible inputs for Equation 5.8 of Sp-1, we look for the lower and upper bounds of the maximal cross-section area,  $a_{max}$ :  $(a_{max})^{min}$  and  $(a_{max})^{max}$ . The lower bound,  $(a_{max})^{min}$ , could be found by a linear programming formulation similar to Equation 5.8:

$$\begin{aligned} & \underset{s_j, a_{max}}{\text{minimize}} && a_{max} \\ & \text{subject to} && (5.8a) - (5.8d). \end{aligned} \tag{5.11}$$

To find the upper bound,  $(a_{max})^{max}$ , we solve Equation 5.8 without constraint 5.8d, and we take the maximal value of  $a_j, j = 1, \dots, |E|$ . Any choice of  $a_{max}$  greater than  $(a_{max})^{max}$  does not activate constraint 5.8d, giving the same solution.

*Main procedure:* Here, we alternate between Sp-1 and Sp-2. The output of Sp-1 includes axial forces,  $s_j$ , which could be directly used as input for Sp-2. However, there is no readily available output from Sp-2 that could be used as input for Sp-1. Therefore, we sample different values of  $a_{max}$  for Sp-1, compute corresponding optimal values, and estimate a gradient to update  $a_{max}$ .

Our approach is illustrated in Figure 5.10: We start the search by  $a_{max}^{0,0} = (a_{max})^{min}$  and  $a_{max}^{1,4} = (a_{max})^{max}$ . At the  $m$ -th iteration, we uniformly sample  $a_{max}^{m,1}$ ,  $a_{max}^{m,2}$ , and  $a_{max}^{m,3}$  to divide the range between  $a_{max}^{m,0}$  and  $a_{max}^{m,4}$ . Among the five samples including the boundaries, we find the  $n$ -th value  $a_{max}^{m,n}$  assuming the smallest optimal volume from Sp-2, and update the range for the next iteration accordingly:  $a_{max}^{k+1,0} = a_{max}^{k,max(0,n-1)}$  and  $a_{max}^{k+1,4} = a_{max}^{k,min(4,n+1)}$ .

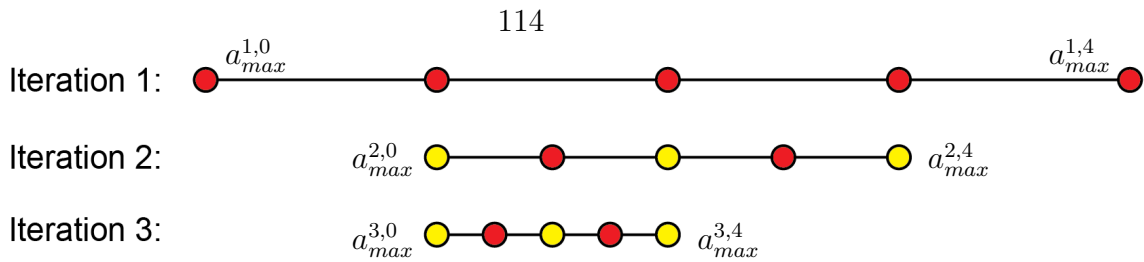


Figure 5.10: We apply Sp-1 and Sp-2 to narrow the range for the best choice of  $a_{max}$ . Starting from  $a_{max}^{1,0} = (a_{max})^{min}$  and  $a_{max}^{1,4} = (a_{max})^{max}$  in Iteration-0, we uniformly sample  $a_{max}^{1,1}$ ,  $a_{max}^{1,2}$ , and  $a_{max}^{1,3}$  to compute the total volumes computed based on both Sp-1 and Sp-2. Then, we update the range according to the minimal value to continue and repeat the procedure. Here, the red dots indicate the values requiring new computation.

*Post-processing:* Finally, we use the computed type assignment from the previous stage to run Sp-3 to further refine the choice of cross-section areas for the beam types.

## 5.5 Results and Discussion

We present example designs, discuss quantitative results, and compare our method with an alternative approach.

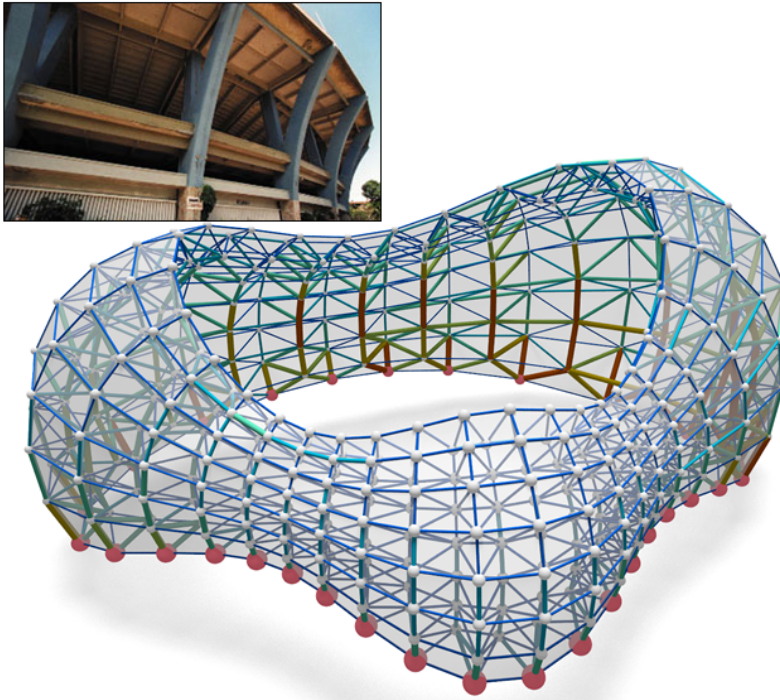


Figure 5.11: Our framework automatically specifies the strong supporting pillars of a quad-based double-layer space structure. The assignment coincides with structural designs observed in real life like the stadium shown in the inset photo.

Structure	Beams	Arbitrary Beam Types Available		1 Customized Beam Type		2 Customized Beam Types		3 Customized Beam Types		6 Customized Beam Types		Fig.
		Vol	T(s)	Vol	T(s)	Vol	T(s)	Vol	T(s)	Vol	T(s)	
Dome	600	42.5	0.06	104.8	0.1	69.6	2.5	59.8	2.6	50.8	2.6	5.7
BM Tri	1200	55.9	0.1	168.7	0.2	104.2	6.3	89.4	7.0	71.2	7.2	5.4
Wave	1680	53.7	0.18	399.0	0.3	141.3	10.0	99.6	10.3	78.3	11.1	5.11
Flat Roof	2048	30.1	0.2	98.6	0.3	56.9	11.9	45.0	12.7	35.8	12.8	5.17
Blob	2842	78.66	0.5	681.3	0.6	272.1	23.4	186.9	25.3	153.4	25.6	5.15
BM Quad	4096	50.7	0.5	277.3	0.8	99.9	31.8	86.1	50.5	66.9	110.8	5.4
BM Hex	5184	86.9	0.6	368.3	1.3	149.9	33.6	128.8	44.4	108.3	50.9	5.4
Lilium	7824	102.3	1.6	493.1	2.8	226.7	79.6	178.9	94.7	139.4	92.3	5.16
Tunnel	10828	46.4	2.0	179.7	3.2	92.8	139.2	75.1	164.9	60.8	157.5	5.18
Flower	12240	70.3	3.3	363.5	22.3	147.6	433.8	117.5	481.6	93.0	504.8	5.12
BM 3464	19584	76.8	5.2	411.5	44.4	167.1	272.7	131.1	377.0	100.3	429.4	5.4
Baku	20768	150.2	7.1	747.5	701.1	1352.8	983.6	269.0	1181.7	209.3	1201.4	??
Train Station	23552	119.3	8.2	458.8	935.3	242.2	1433.6	194.7	1536.3	157.2	1646.9	5.2

Table 5.1: Quantitative analysis of examples (sorted according to the number of beams). For each example, we show the number of beams, computational time and total volume for constructing solutions using 1, 2, 3, 6, and arbitrary types of beams. Increasing the number of beam types reduces the total volume, at the cost of increased manufacturing complexity and slightly more computational time.

**Example Designs** We illustrate construction solutions for a double-layer space structure design motivated by a real project in Figures ?? and 5.14. We present a design of the train station model in Figure 5.2, and space structures with base meshes of different connectivities in Figure 5.4. In addition, we show a result resembling a real construction in Figure 5.11, a freeform pillar in Figures 5.12 and 5.13, a triangular single layer space structure in Figure 5.15, a structure approximating the Lilium Tower model in Figure 5.16, a roof spanning a square domain in Figure 5.17, and a tunnel based on a semi-regular pattern in Figure 5.18.

**Quantitative Evaluation** Our framework is implemented in C++ with customized

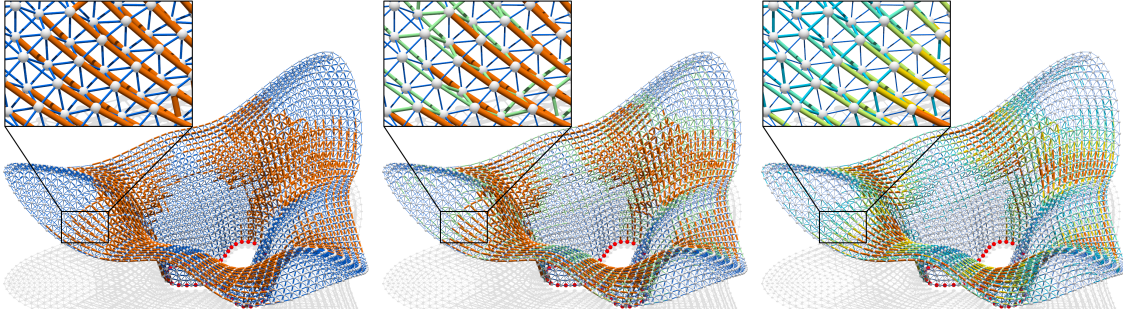


Figure 5.12: From left to right: three construction solutions with 2, 3 and 6 types of customized beams for the flower model featuring an extending freeform roof. The thicker beams are adaptively assigned to regions with strong bending moments, as shown in Figure 5.13.

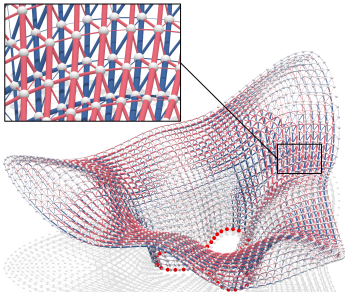


Figure 5.13: Color coding of tension (purple) and compression (blue) for the construction solution on the right side of Figure 5.12. The strong bending moments are transferred to tensile and compressive axial forces along the beams within the two layers.

data structures. We use the HLBFGS Library developed in [64] for the multi-objective nonlinear optimization procedure and Mosek for linear programming. We run our tests on a workstation with an Intel Xeon X5550 2.67GHz processor. In Table 5.1, we report the computational time and achieved total volume for each model when the number of customized beam types is 1, 2, 3, 6, and arbitrary.

**Comparisons** A straightforward and seemingly standard alternative to our volume

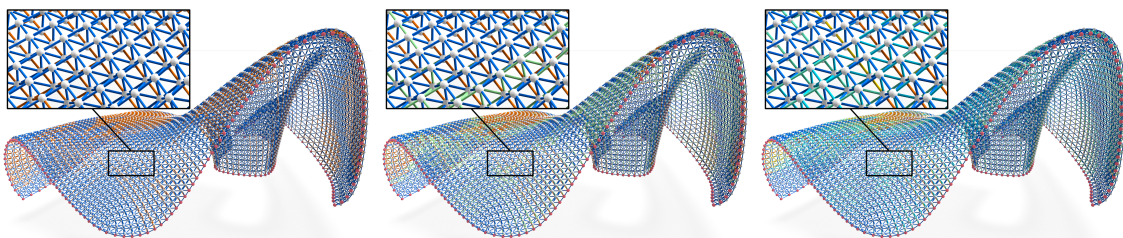


Figure 5.14: From left to right: three construction plans using 2, 3 and 6 types of customized beams for a statically sound space structure designed and optimized with our framework, motivated by the real project shown in Figure 5.1. The solution on the right is illustrated in Figure ??.

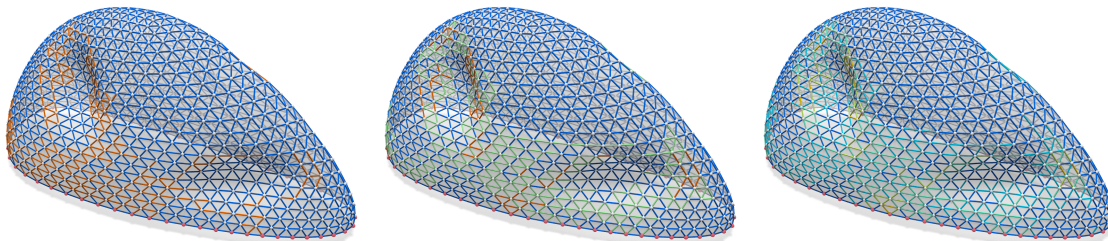


Figure 5.15: Construction solutions with 2, 3, and 6 types of beams for a single layer triangular space structure approximating the Blob model, following a real architectural project in Eindhoven, the Netherlands. Our method automatically adjusts beam cross-section areas adaptively, and thicker beams are used near the dent where stresses are concentrated to ensure the structural soundness.

Structure	Beams	Beam Types	Ours		MILP	
			Vol	T(s)	Vol	T(s)
Small	101	2	15.96	0.83	17.44	0.81
		6	10.19	0.87	10.96	55.20
Dome Fig. 5.7	600	2	69.59	2.58	70.26	305
		6	50.81	2.64	51.97	620
Lilium Fig. 5.16	7824	2	226.17	79.57	260.17	610
		6	139.71	92.32	190.16	30h

Table 5.2: Comparison to mixed integer programming. We show results for three models and two configurations of each model with 2 and 6 beam types. We list the number of beams and compare our volume and running time with the mixed integer programming solutions.

optimization method described in Subsection 5.4.4 is to alternate solving the non-relaxed form of Sp-1 with Sp-3. In Table 5.2, we compare our approach to this alternative with Mosek as the mixed-integer linear programming (MILP) solver for Sp-1. Our method provides construction solutions with less volume at a much faster speed and exhibits better scalability. The key problem for mixed integer programming is the initialization. Without a global search strategy, as present in our method, mixed integer programming converges to an undesirable local minimum, despite its very high computation time.

**Limitations** A major limitation of our framework is the fixed connectivity during optimization phases discussed in Subsections 5.4.3 and 5.4.4. While we provide a set of connectivity initialization tools, it would be desirable to suggest automatic connec-

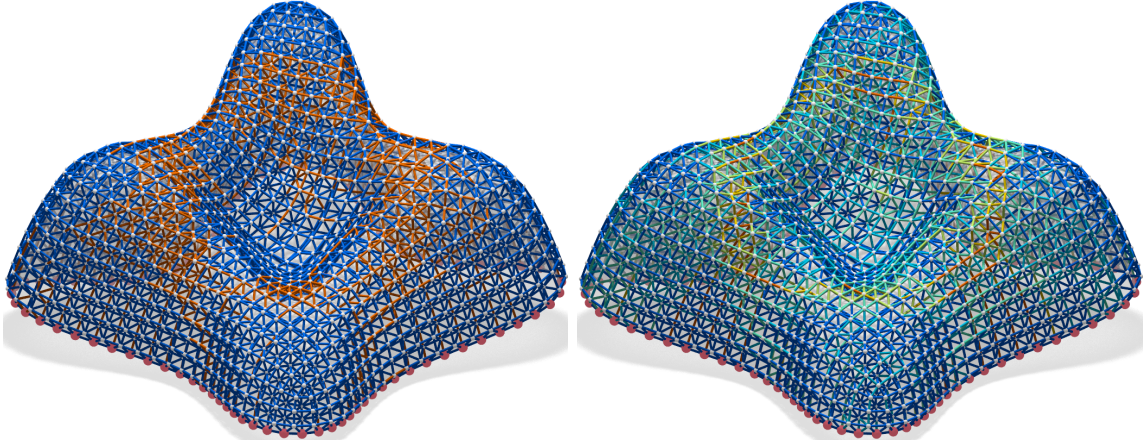


Figure 5.16: Construction solutions of a double-layer space structure based on a quadrilateral mesh approximating the Lilium Tower model with 2 (left) and 6 (right) types of customized beams.

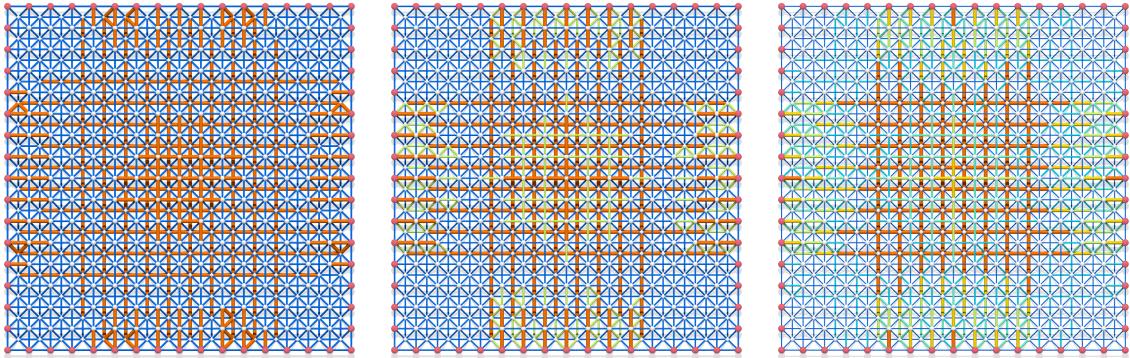


Figure 5.17: Constructing a symmetric and flat roof spanning a square boundary with 2 (left), 3 (middle) and 6 (types) of beams.

tivity changes during the nonlinear and discrete optimization procedures. Another major limitation of our work is the lack of control for global buckling. We have not considered the scenario that the overall structure as whole fails while each beam is still statically sound. In such a case, the space structure overall, instead of a single beam, might lack the stiffness, due to the existence of small eigenvalues in the load-stiffness matrix. This is also a common limitation for other static-aware computational design tools for initial stages, e.g., for self-supporting masonry structures.

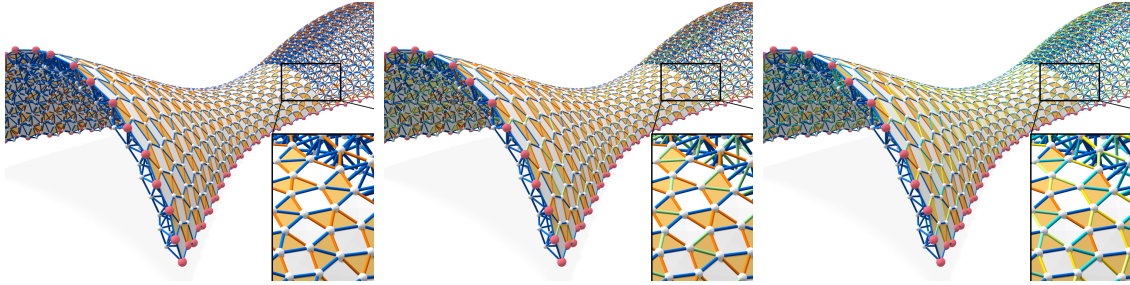


Figure 5.18: Construction solutions with 2, 3, and 6 types of beams for a space structure approximating a freeform tunnel. The base mesh is a semi-regular pattern consisting of triangles and quadrilaterals.

## 5.6 Conclusion

We study the design and volume optimization of space structures. Our computational framework creates space structures in static equilibrium with reduced material usage. Moreover, we incorporate the practical consideration for construction complexity where the beam types are limited and solve this challenging problem in a practical and efficient manner.

# Chapter 6

## Conclusion

### 6.1 Summary

Geometric rationalization as a core research problem of architectural geometry has been studied in this thesis through four specific example problems inspired by real life applications. They include (1) shading and lighting systems, (2) freeform honeycomb structures, (3) polyhedral patterns, and (4) space frame structures.

The principal goal of geometric rationalization is to reduce the difficulty of manufacturing and assembling of components of freeform architecture with complex geometry. The four example problems highlight different construction requirements such as

- flat faces, either beams or panels (e.g., Chapter 2, 3 and 4).
- torsion-free nodes (e.g., Chapter 2 and 3).
- congruent nodes (e.g., Chapter 3).
- repetitive bars (e.g., Chapter 5).

We achieve our results by developing efficient optimization algorithms with aesthetic and functional constraints and theoretical studies of the underlying geometry.



## 6.2 Future Work

For future work, we would like to explore other types of geometric rationalization formulations and we also wish to broaden the scope of the applications to other fields in geometry processing to explore if similar techniques can be applied there. Example applications are as follows.

**Dynamic shading and lighting system.** The shading and lighting systems shown in chapter 2 are all static so that they cannot adaptively change their structures according to different sunlight directions. Usually, static shading and lighting systems are optimized to perform well for a certain sunlight condition. For example, a static shading system could be optimized to block the sunlight at the hottest time. However, the actual sunlight direction changes all the time. When the sunlight direction differs too much from the one being optimized, the performance of static systems will be suboptimal. Because of this, a dynamic shading or lighting system is desirable to achieve the functional goal by deforming its structure according to different sunlight directions.

**Packing problems and geometry optimization.** Flat elements, either panels or beams, are the easiest and cheapest to produce. Usually, they are manufactured by cutting from large flat sheets of glass or metal. A freeform surface or structure consists of a large number of panels and beams with various sizes and shapes. Packing these elements efficiently may be essential for material saving. Meanwhile, the cost of cutting could also be incorporated in this challenging problem. Geometry optimization of the surfaces or structure potentially provides more degrees of freedom for packing these elements when their shapes are considered as variables.

**Topology optimization of space structures.** The geometric rationalization problem in chapter 5 regarding limited types of cross-section areas for the struts of space structures is a challenging mix-integer optimization problem. We assume the topology of space frames to be fixed in the optimization. If we allow the connectivities

to change by adding or deleting nodes and struts, we will obtain more degrees of freedom to compute the space structure with less volume and better stability. The combination of topology optimization, volume optimization with the constraints of force equilibrium is interesting and challenging.

**Truss-like supporting structures for 3D printing.** Due to their flexibility, stability, and cost-effectiveness, space frame structures are popularly used as the support structures for industrial workshops, train stations, airports, and stadiums in the field of architecture and structural engineering. One possible research is to extend the application of space frame structures to 3D printing, specifically, as the support structures for the overhanging areas of printed objects. 3D printing as a general technology has enormous potential in digital fabrication, for both large-scale constructions like buildings and small-scale objects such as those created by 3D nanoprinting. Usually to print a sturdy object, we need supporting structures, especially for overhanging area. Generating statically-sound and material-efficient truss-like support structures to support 3D printed objects is challenging from both sides of geometry processing and digital fabrication. As the materials are added layer by layer, the force at each supporting point due to the gravity is dynamically changed. We need to combine the geometric analysis of 3D models during manufacturing with the design and optimization of their underlying support structures.

# REFERENCES

- [1] H. Pottmann, A. Asperl, M. Hofer, and A. Kilian, *Architectural Geometry*. Bentley Institute Press, 2007.
- [2] H. Pottmann, M. Eigensatz, A. Vaxman, and J. Wallner, “Architectural geometry,” *Computers & Graphics*, vol. 47, pp. 145–164, 2015.
- [3] I. R. Porteous, *Geometric Differentiation for the Intelligence of Curves and Surfaces*. Cambridge Univ. Press, 1994.
- [4] H. Pottmann and J. Wallner, *Computational Line Geometry*. Springer, 2001.
- [5] Q. J. Ge and B. Ravani, “Geometric design of rational Bezier line congruences and ruled surfaces using line geometry,” *Computing [Suppl.]*, vol. 13, pp. 101–120, 1998.
- [6] J. Yu, X. Yin, X. Gu, L. McMillan, and S. Gortler, “Focal surfaces of discrete geometry,” in *Proc. SGP*, 2007, pp. 23–32.
- [7] A. Bobenko and Yu. Suris, *Discrete Differential Geometry: Integrable Structure*. American Math. Soc., 2009.
- [8] A. Doliwa, P. Santini, and M. Mañas, “Transformations of quadrilateral lattices,” *J. Math. Phys.*, vol. 41, pp. 944–990, 2000.
- [9] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang, “Geometric modeling with conical meshes and developable surfaces,” *ACM Trans. Graphics*, vol. 25, no. 3, pp. 681–689, 2006.
- [10] H. Pottmann, Y. Liu, J. Wallner, A. Bobenko, and W. Wang, “Geometry of multi-layer freeform structures for architecture,” *ACM Trans. Graphics*, vol. 26, no. 3, pp. # 65,1–11, 2007.

- [11] H. Pottmann and J. Wallner, “The focal geometry of circular and conical meshes,” *Adv. Comp. Math*, vol. 29, pp. 249–268, 2008.
- [12] H. Pottmann, A. Schiftner, P. Bo, H. Schmiedhofer, W. Wang, N. Baldassini, and J. Wallner, “Freeform surfaces from single curved panels,” *ACM Trans. Graph.*, vol. 27, no. 3, pp. #76,1–10, 2008, Proc. SIGGRAPH.
- [13] T. Kiser, M. Eigensatz, M. M. Nguyen, P. Bompas, and M. Pauly, “Architectural caustics – controlling light with geometry,” in *Advances in Architectural Geometry 2012*, L. Hesselgren *et al.*, Eds. Springer, 2012, pp. 91–106.
- [14] D. C. Liu and J. Nocedal, “On the limited memory method for large scale optimization,” *Math. Prog. B*, vol. 45, pp. 503–528, 1989.
- [15] Y. Liu, W. Xu, J. Wang, L. Zhu, B. Guo, F. Chen, and G. Wang, “General planar quadrilateral mesh design using conjugate direction field,” *ACM Trans. Graph.*, vol. 30, pp. #140, 1–10, 2011, Proc. SIGGRAPH Asia.
- [16] D. Bommers, H. Zimmer, and L. Kobbelt, “Mixed-integer quadrangulation,” *ACM Trans. Graphics*, vol. 28, no. 3, pp. #77, 1–10, 2009.
- [17] L. Lu, A. Sharf, H. Zhao, Y. Wei, Q. Fan, X. Chen, Y. Savoye, C. Tu, D. Cohen-Or, and B. Chen, “Build-to-last: Strength to weight 3D printed objects,” *ACM Trans. Graph.*, vol. 33, no. 4, 2014, proc. SIGGRAPH.
- [18] H. Pottmann, Y. Liu, J. Wallner, A. Bobenko, and W. Wang, “Geometry of multi-layer freeform structures for architecture,” *ACM Trans. Graph.*, vol. 26, pp. #65,1–11, 2007, Proc. SIGGRAPH.
- [19] J. Wang, C. Jiang, P. Bompas, J. Wallner, and H. Pottmann, “Discrete line congruences for shading and lighting,” *Comput. Graph. Forum*, vol. 32, no. 5, pp. 53–62, 2013, proc. SGP.
- [20] A. Schiftner, M. Höbinger, J. Wallner, and H. Pottmann, “Packing circles and spheres on surfaces,” *ACM Trans. Graph.*, vol. 28, no. 5, pp. #139,1–8, 2009, Proc. SIGGRAPH Asia.

- [21] H. Pottmann, C. Jiang, M. Höbinger, J. Wang, P. Bompas, and J. Wallner, “Cell packing structures,” *Computer-Aided Design*, vol. 50, pp. 70–83, 2015, to appear.
- [22] A. Schiftner, M. Eigensatz, M. Kilian, and G. Chinzi, “Large scale double curved glass facades made feasible – the Arena Corinthians west facade,” in *Glass Performance Days Finland (Conference Proceedings)*, 2013, pp. 494 – 498.
- [23] M. Singh and S. Schaefer, “Triangle surfaces with discrete equivalence classes,” *ACM Trans. Graph.*, vol. 29, pp. #46,1–7, 2010, Proc. SIGGRAPH.
- [24] C.-W. Fu, C.-F. Lai, Y. He, and D. Cohen-Or, “K-set tilable surfaces,” *ACM Trans. Graph.*, vol. 29, pp. #44,1–6, 2010, Proc. SIGGRAPH.
- [25] P. Bo, H. Pottmann, M. Kilian, W. Wang, and J. Wallner, “Circular arc structures,” *ACM Trans. Graph.*, vol. 30, pp. #101,1–11, 2011.
- [26] H. Zimmer, M. Campen, D. Bommès, and L. Kobbelt, “Rationalization of triangle-based point-folding structures,” *Comput. Graph. Forum*, vol. 31, pp. 611–620, 2012, proc. Eurographics.
- [27] M. Nieser, J. Palacios, K. Polthier, and E. Zhang, “Hexagonal global parameterization of arbitrary surfaces,” *IEEE Trans. Vis. Comp. Graphics*, vol. 18, no. 6, pp. 865–878, 2012.
- [28] D. Bommès, H. Zimmer, and L. Kobbelt, “Mixed-integer quadrangulation,” *ACM Trans. Graph.*, vol. 28, no. 3, pp. #77,1–10, 2009, Proc. SIGGRAPH.
- [29] C. Tang, X. Sun, A. Gomes, J. Wallner, and H. Pottmann, “Form-finding with polyhedral meshes made simple,” *ACM Trans. Graphics*, vol. 33, no. 4, 2014, Proc. SIGGRAPH.
- [30] M. Zadavec, A. Schiftner, and J. Wallner, “Designing quad-dominant meshes with planar faces,” *Comput. Graph. Forum*, vol. 29, no. 5, pp. 1671–1679, 2010, Proc. SGP.

- [31] P. Song, C.-W. Fu, P. Goswami, J. Zheng, N. Mitra, and D. Cohen-Or, “Reciprocal frame structures made easy,” *ACM Trans. Graph.*, vol. 32, no. 4, pp. #94,1–10, 2013, Proc. SIGGRAPH.
- [32] S. J. Abas, A. S. Salman, A. Moustafa, and M. Atiyah, *Symmetries of Islamic geometrical patterns*. World Scientific, 1995, vol. 3.
- [33] P. J. Lu and P. J. Steinhardt, “Decagonal and quasi-crystalline tilings in medieval islamic architecture,” *Science*, vol. 315, no. 5815, pp. 1106–1110, 2007.
- [34] O. D. Krieg, T. Schwinn, A. Menges, J.-M. Li, J. Knippers, A. Schmitt, and V. Schwieger, “Biomimetic lightweight timber plate shells: Computational integration of robotic fabrication, architectural geometry and structural design,” in *Advances in Architectural Geometry 2014*. Springer, 2015, pp. 109–125.
- [35] D. Cohen-Steiner, P. Alliez, and M. Desbrun, “Variational shape approximation,” *ACM Trans. Graphics*, vol. 23, no. 3, pp. 905–914, 2004, proc. SIGGRAPH.
- [36] R. Sauer, *Differenzengeometrie*. Springer, 1970.
- [37] A. Bobenko and Yu. Suris, *Discrete differential geometry: Integrable Structure*. American Math. Soc., 2008.
- [38] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang, “Geometric modeling with conical meshes and developable surfaces,” *ACM Trans. Graph.*, vol. 25, no. 3, pp. 681–689, 2006, Proc. SIGGRAPH.
- [39] H. Zimmer, “Optimization of 3D models for fabrication,” Ph.D. dissertation, RWTH Aachen, 2014.
- [40] Y. Li, Y. Liu, and W. Wang, “Planar hexagonal meshing for architecture,” *IEEE Trans. Vis. Comp. Graphics*, vol. 21, pp. 95–106, 2014.
- [41] A. Vaxman and M. Ben-Chen, “Dupin meshing: A parameterization approach to planar hex-dominant meshing,” Technion, Tech. Rep. CS-2015-01 (CS series), 2015.

- [42] C. Jiang, J. Wang, J. Wallner, and H. Pottmann, “Freeform honeycomb structures,” *Comput. Graph. Forum*, vol. 33, no. 5, pp. 185–194, 2014, proc. SGP.
- [43] R. Poranne, E. Ovreiu, and C. Gotsman, “Interactive planarization and optimization of 3D meshes,” *Comput. Graph. Forum*, vol. 32, no. 1, pp. 152–163, 2013.
- [44] S. Bouaziz, Y. Schwartzburg, T. Weise, and M. Pauly, “Shaping discrete geometry with projections,” *Computer Grap. Forum*, vol. 31, pp. 1657–1667, 2012, proc. SGP.
- [45] B. Deng, S. Bouaziz, M. Deuss, A. Kaspar, Y. Schwartzburg, and M. Pauly, “Interactive design exploration for constrained meshes,” *Computer-Aided Design*, vol. 61, pp. 13–23, 2015.
- [46] M. Huard, P. Bompas, and M. Eigensatz, “Planar panelization with extreme repetition,” in *Advances in Architectural Geometry 2014*, P. Block *et al.*, Eds. Springer, 2014.
- [47] E. Akleman, V. Srinivasan, and E. Mandal, “Remeshing schemes for semi-regular tilings,” in *Proceedings of the International Conference on Shape Modeling and Applications 2005*, 2005, pp. 44–50.
- [48] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt, “Openmesh - a generic and efficient polygon mesh data structure,” 2002.
- [49] S. Toledo, “TAUCS, a library of sparse linear solvers,” C library, <http://www.tau.ac.il/~stoledo/taucs/>, 2003.
- [50] A. G. M. Michell, “Lviii. the limits of economy of material in frame-structures,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 8, no. 47, pp. 589–597, 1904.
- [51] W. S. Dorn, “Automatic design of optimal structures,” *Journal de mecanique*, vol. 3, pp. 25–52, 1964.

- [52] T. Zegard and G. H. Paulino, “Grandground structure based topology optimization for arbitrary 2d domains using matlab,” *Structural and Multidisciplinary Optimization*, vol. 50, no. 5, pp. 861–882, 2014.
- [53] T. Mitchell, “A limit of economy of material in shell structures,” Ph.D. dissertation, University of California, Berkeley, 2013.
- [54] J. Smith, J. Hodgins, I. Oppenheim, and A. Witkin, “Creating models of truss structures with optimization,” in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 295–301.
- [55] W. Wang, T. Y. Wang, Z. Yang, L. Liu, X. Tong, W. Tong, J. Deng, F. Chen, and X. Liu, “Cost-effective printing of 3d objects with skin-frame structures,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 177, 2013.
- [56] P. Block, “Thrust network analysis: Exploring three-dimensional equilibrium,” Ph.D. dissertation, M.I.T., 2009.
- [57] Y. Liu, H. Pan, J. Snyder, W. Wang, and B. Guo, “Computing self-supporting surfaces by regular triangulation,” *ACM Trans. Graph.*, vol. 32, no. 4, pp. #92,1–10, 2013, proc. SIGGRAPH.
- [58] D. Panozzo, P. Block, and O. Sorkine-Hornung, “Designing unreinforced masonry models,” *ACM Trans. Graph.*, vol. 32, no. 4, pp. #91,1–12, 2013, Proc. SIGGRAPH.
- [59] F. de Goes, P. Alliez, H. Owhadi, and M. Desbrun, “On the equilibrium of simplicial masonry structures,” *ACM Trans. Graph.*, vol. 32, no. 4, pp. #93,1–10, 2013, Proc. SIGGRAPH.
- [60] E. Vouga, M. Höbinger, J. Wallner, and H. Pottmann, “Design of self-supporting surfaces,” *ACM Trans. Graph.*, vol. 31, no. 4, pp. #87,1–11, 2012, proc. SIGGRAPH.
- [61] C.-W. Fu, C.-F. Lai, Y. He, and D. Cohen-Or, “K-set tilable surfaces,” in *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4. ACM, 2010, p. 44.



- [62] C. Jiang, C. Tang, M. Tomičić, J. Wallner, and H. Pottmann, “Interactive modeling of architectural freeform structures – combining geometry with fabrication and statics,” in *Advances in Architectural Geometry 2014*, P. Block *et al.*, Eds. Springer, 2014.
- [63] C.-H. Peng, M. Barton, C. Jiang, and P. Wonka, “Exploring quadrangulations,” *ACM Trans. Graph.*, vol. 33, pp. #12,1–13, 2014.
- [64] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, “On centroidal voronoi tessellation energy smoothness and fast computation,” *ACM Transactions on Graphics (ToG)*, vol. 28, no. 4, p. 101, 2009.
- [65] C. Jiang, C. Tang, A. Vaxman, P. Wonka, and H. Pottmann, “Polyhedral patterns,” *ACM Trans. on Graphics*, vol. 34, no. 6, pp. #172, 1–12, 2015, Proc. SIGGRAPH Asia.
- [66] W.-F. Chen and E. M. Lui, *Handbook of structural engineering*. Crc Press, 2005.
- [67] D. C. Liu and J. Nocedal, “On the limited memory bfgs method for large scale optimization,” *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.
- [68] S. Agmon, “The relaxation method for linear inequalities,” *Canadian Journal of Mathematics*, vol. 6, no. 3, pp. 382–392, 1954.

# APPENDICES

## A Papers Published and Under Preparation

- **Caigui Jiang**, Chengcheng Tang, Peter Wonka, “Design and Volume Optimization of Space Structures”, *Submitted to SIGGRAPH Asia 2016*.
- **Caigui Jiang**, Felix Günther, Johannes Wallner, Helmut Pottmann, “Measuring and controlling fairness of triangulations”, *Advances in Architectural Geometry (to appear)*, 2016.
- Xiang Sun, **Caigui Jiang**, Johannes Wallner, Helmut Pottmann, “Vertex Normals and Face Curvatures of Triangle Meshes”, *Advances in Discrete Differential Geometry, Springer (to appear)*, 2016.
- **Caigui Jiang**, Chengcheng Tang, Amir Vaxmann, Peter Wonka, Helmut Pottmann, “Polyhedral Patterns”, *ACM Transactions on Graphics, SIGGRAPH Asia*, 2015.
- **Caigui Jiang**, Chengcheng Tang, Jun Wang, Johannes Wallner, Helmut Pottmann, “Freeform Honeycomb Structures and Lobel Frames”, *SIGGRAPH Posters*, 2015.
- Liangliang Nan, **Caigui Jiang**, Bernard Ghanem, Peter Wonka, “Template Assembly for Detailed Urban Reconstruction”, *Computer Graphics Forum(Proc. EG 2015)*, 2015.

- Helmut Pottmann, **Caigui Jiang**, Mathias Hbinger, Jun Wang, Philippe Bompas, and Johannes Wallner, “Cell Packing Structures”, *Computer-Aided Design*, 2015.
- **Caigui Jiang**, Chengcheng Tang, Marko Tomicic, Johannes Wallner, Helmut Pottmann, “Interactive Modeling of Architectural Freeform Structures,” , *Advances in Architectural Geometry*, 2014.
- **Caigui Jiang**, Jun Wang, Johannes Wallner, Helmut Pottmann, “Freeform Honeycomb Structures”, *Computer Graphics Forum(Proc. SGP 2014)*, 2014.
- Chi-Han Peng, Michael Barton, **Caigui Jiang**, and Peter Wonka, “ Exploring Quadrangulations”, *ACM Transactions on Graphics* , 2014.
- **Caigui Jiang**, Jun Wang, Philippe Bompas, Johannes Wallner, Helmut Pottmann, “Freeform Shading and Lighting Systems from Planar Quads” , *Design Modelling Symposium Berlin* , 2013.
- Jun Wang, **Caigui Jiang**, Philippe Bompas, Johannes Wallner, Helmut Pottmann, “Discrete Line Congruences for Shading and Lighting” , *Computer Graphics Forum(Proc. SGP 2013)*, 2013.