Design with Nonlinear Constraints

Thesis by

Chengcheng Tang

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

December, 2015

The thesis of Chengcheng Tang is approved by the examination committee

Committee Chairperson:	Helmut Pottmann
Committee Co-Chairperson:	Peter Wonka
Committee Member:	Peter Markowich
Committee Member:	Mark Pauly

King Abdullah University of Science and Technology

2015

ABSTRACT

Design with Nonlinear Constraints

Chengcheng Tang

Most modern industrial and architectural designs need to satisfy the requirements of their targeted performance and respect the limitations of available fabrication technologies. At the same time, they should reflect the artistic considerations and personal taste of the designers, which cannot be simply formulated as optimization goals with single best solutions. This thesis aims at a general, flexible yet efficient computational framework for interactive creation, exploration and discovery of serviceable, constructible, and stylish designs. By formulating nonlinear engineering considerations as linear or quadratic expressions by introducing auxiliary variables, the constrained space could be efficiently accessed by the proposed algorithm *Guided Projection*, with the guidance of aesthetic formulations. The approach is introduced through applications in different scenarios. Its effectiveness is demonstrated by examples that were difficult or even impossible to be computationally designed before. The first application is the design of meshes under both geometric and static constraints, including self-supporting polyhedral meshes that are not height fields. Then, with a formulation bridging mesh-based and spline-based representations, the application is extended to developable surfaces including origami with curved creases. Finally, general approaches to extend hard constraints and soft energies are discussed, followed by a concluding remark outlooking possible future studies.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Helmut Pottmann. He taught me how to ask and concentrate on the right questions and solve problems in a focused manner. His geometric intuitions, research insights, and artistic tastes helped me to develop my appreciation towards the beauty of research. His excellent lectures showed me how to share and present my knowledge and understandings. Discussions and conversations with Helmut are always great sources of inspiration for both research and life. From many perspectives, I am extremely lucky to be a student of Helmut.

I would like to thank my co-advisor Peter Wonka. His energy, passion and enthusiasm about research have always been motivating. His exceptional vision of problems, projects, and careers helped me to develop a bigger picture. I am grateful to Peter Markowich and Mark Pauly for joining my thesis committee. The multiple courses lectured by Peter have significantly deepened my mathematical understandings, and the suggestions from Mark are truly valuable for both the thesis and my future work.

Furthermore, I would like to thank Johannes Wallner for being a great collaborator on multiple projects. He is also an excellent mentor who has greatly helped my research and a good friend who shares many interests. I am grateful to Niloy Mitra and Amir Vaxman, for the enjoyable and exciting collaborations and discussions. I would also like to thank my other collaborators, including Pengbo Bo, Caigui Jiang, Yongliang Yang, Xin Zhao, Xiang Sun, Alexandra Aguiar Gomes, and Marko Tomičić.

The Visual Computing Center has been a very supportive family. I am truly grateful to all its members, including Yanchao Yang, Han Liu, Youyi Zheng, Qiang Fu, Dongming Yan, Lubin Fan, Liliana Rivera, Tina Smith, Naeemullah Khan, Markus

Hadwiger, Rachid Ait-Haddou, Ganesh Sundaramoorthi, Bernard Ghanem, Wolfgang Heidrich, Liangliang Nan, Antoine Vigneron, Rahman Hasan, Minglei Li, Haiyong Jiang, Baoyuan Wu, Daniel Binham, Neil Smith, Mohamed Shalaby, Ronell Sicat, Jens Schneider, Ling Shi, Jun Wang, Johanna Beyer, Wito Engelke, Lei Xiao, Robin Swanson, Yifan Peng, Shuocheng Su, Zhaojin Lu, Jinlei Wang, and Jiliang Li.

I appreciate the academic and spiritual support from many KAUST faculty members, including Gerard Schuster, Chris Gehring, Mohamed-Slim Alouini, Raul Tempone, David Ketcheson, Aslan Kasimov, David Keyes, Xiangliang Zhang, Ying Wu, Yu Han, Georgiy Stenchikov, Hakan Bagci, Peng Wang, Boon Ooi, Mootaz Elnozahy, Suk Ho Chung, Xixiang Zhang, Sigurdur Thoroddsen, and Sigurjon Jonsson. I thank Jill Pagels, Naseer Aijaz, Hebatalla Mahmoud, and Aida Hoteit for their great help.

I thank my KAUST friends for the wonderful journey together, including Felix Lau, Yangqin Gao, Buyi Yan, Daquan Guo, Peng Zhan, Ting Zhang, Le Shi, Batian Chen, Yuanfang Hou, Chao Shen, Zhongwei Wang, Kai Lu, Ying Yi, Fuquan Li, Xiaolei Wang, Nail Ussembayev, Jesus Sierra, Andrew Yip, Yili He, Aloysius Wong, Dmitry Kabanov, Shanran Tang, Jiaming Zhang, Sheng Li, Wenwen Zhu, Yiqiang Fan, Bodong Li, Ruogu Ding, Chengbin Peng, Yi Ren, Lisong Xu, Yang Liu, Julia Chin, Jinkai Xue, Wengang Li, Weigang Li, Hao Ma, Abdullah Ali Abdullah, Junsong Zhao, Marie-Jean Thoraval, Junjie Xiong, Anlei Rao, Keyang Dai, Wei Chen, Jian Sun, Ke Feng, Song Feng, Zhikao Li, David Evans, Jiangjiang Pan, and Longfei Gao.

I deeply thank my father, Shaoming Tang, and my mother, Zhongqiong Chen. With an enormous amount of quality time spent with me during my growth, they are my first and best teachers. Filled with love and optimism, they have always been encouraging and supportive of me for my pursuit of curiosity and adventure.

Finally, with my deepest love, I would like to thank my wife, Ye Han. As my best friend and companion, her love, encouragement, understanding, patience, humor, stories, and poems are the greatest support and reward for my work and life.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Modern computational design tools have helped designers to convert their ideas to designs efficiently. To further ensure the functionality of the designs, and to fabricate them with available technologies, many restrictions have to be respected. For example, in modern freeform architectural designs, stylish doubly curved buildings have to meet desired performance requirements, and be constructed with simple and cost-effective methods. However, such a task is usually challenging, and it happens often that a brilliant design might not work or can not be built, and one ends up with a mediocre solution after many rounds of trial and error.

By incorporating restrictions while respecting aesthetics, this thesis aims to provide a computational platform for the creation and exploration of artistic architectural and industrial designs that could be engineered and fabricated. The proposed approach is demonstrated in different applications in the following chapters.

1.1 Form-finding with polyhedral meshes

This chapter introduces *Guided Projection* through form-finding with polyhedral meshes. Polyhedral meshes are meshes with planar faces, which are important for the construction of freeform surfaces, as planar panels can be easily manufactured.

Special types of polyhedral meshes, such as circular meshes where each quadrilateral face has a circumcircle, are important for the construction of multi-layer structures. Besides geometric constraints, physical constraints such as static equilibrium under appropriate material assumptions are also important for the servicebility of buildings. To satisfy these considerations, this chapter demonstrates the application of the proposed approach to freeform architectural designs under both geometric and physical constraints.

1.2 Interactive design with developable surfaces

This chapter extends the hard constraints of *Guided Projection* from the planarity of faces to the developability of surfaces. Developable surfaces are surfaces that could be flattened to a plane without tearing, shearing or stretching. As they could be created from sheet materials by bending and welding, they can also be easily manufactured. Thus, they have been widely applied in industry and architecture, such as ancient sails and hulls as well as several modern freeform architectural projects. Developable surfaces also serve as the basic elements of several art creations like origami. To provide an interactive interface for the design of developable surfaces, this chapter extends the proposed approach for the design of multi-patch B-Spline developable surfaces. This chapter also studies relevant properties including special boundary conditions such as developable curved creases, and isometric manipulation of developable surfaces.

1.3 Further Extensions of *Guided Projection*

This chapter exemplifies more scenarios to demonstrate general procedures to extend *Guided Projection*. For the extension of hard constraints defining the design spaces, creating structures with repetitive elements is demonstrated. Besides, generalizing soft energies serving as regularizers of the computation is exemplified through polyhedral patterns, which are meshes with planar faces arranged regularly. To find the feasible regularity of polyhedral patterns, this chapter discusses the explicit constructions approximating surfaces with different Gaussian curvatures, leading to affine symmetry based regularizers applicable in the general case.

1.4 Conclusion and Outlook

After examining the framework of *Guided Projection*, including the hard constraints, the guiding soft energies, the applications and several extensions, concluding remarks are made in this chapter. It also provides a discussion of the future work and possible further applications extending those mentioned in this thesis for general computational design approaches.

Chapter 2

Formfinding with Polyhedral Meshes

2.1 Introduction

The advance of computer aided design platforms and shape modeling systems have provided designers and architects powerful tools to create and shape digital forms and structures for industrial and architectural designs. While most of the available computational solutions are well suited to handle shapes and appearances in solely digital representations, much less could be said for the fabrication and functionality. Therefore, the conceptualization of shapes and forms are decoupled from engineering and manufacturing in most situations. Often, to bring industrial and architectural designs to the real world with expectations of service criteria, many iterations of concepts to validations have to be taken through trial and error.

More recently, there is a trend in computational design to incorporate considerations related fabrication and functionality into the early phase of digital design, facilitating the exploration of valid concepts. A major beneficiary of this movement is the design of freeform architecture. In that field, the progress of fabrication-aware

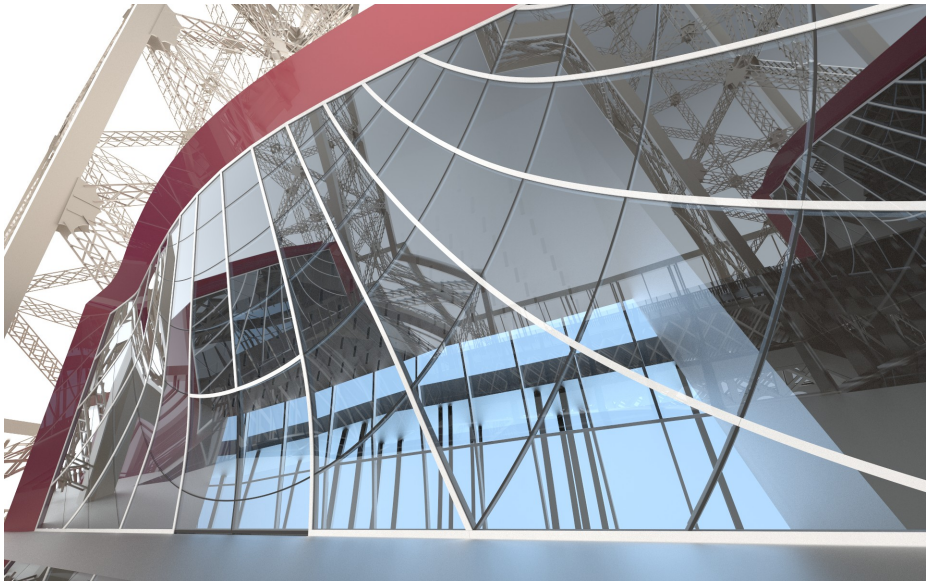


Figure 2.1: The project of Eiffel Tower Pavilions finished in 2014 features doubly curved surfaces. Geometric computation has been applied here for the construction of the facades from cylindrical glass panels, as well as the manufacture of curved steel beams by cutting, bending and welding planar steel sheets. More details could be found in [EKS+10] and [SLB+12].

design has opened up a new research direction called architectural geometry. Fundamental questions asked in architectural geometry include, how to tile a freeform surface with planar panels or developable patches, how to assemble multi-layer structures with beams and nodes, and how to create statically sound vaults with given materials such as stone or timber. Many research results in this field have been successfully applied to a variety of modern landmark architectural projects. Examples include the Eiffel Tower Pavilions in Paris shown in Figure 2.1 and the MLK Jr. Park Stone Vault in Austin shown in Figure 2.2.

The success of those applications further requires a design framework for architects to interactively explore meshes or structures that satisfy both geometric and physical constraints. Equivalently, the goal is to incorporate more fabrication and engineering related considerations into the design phase.

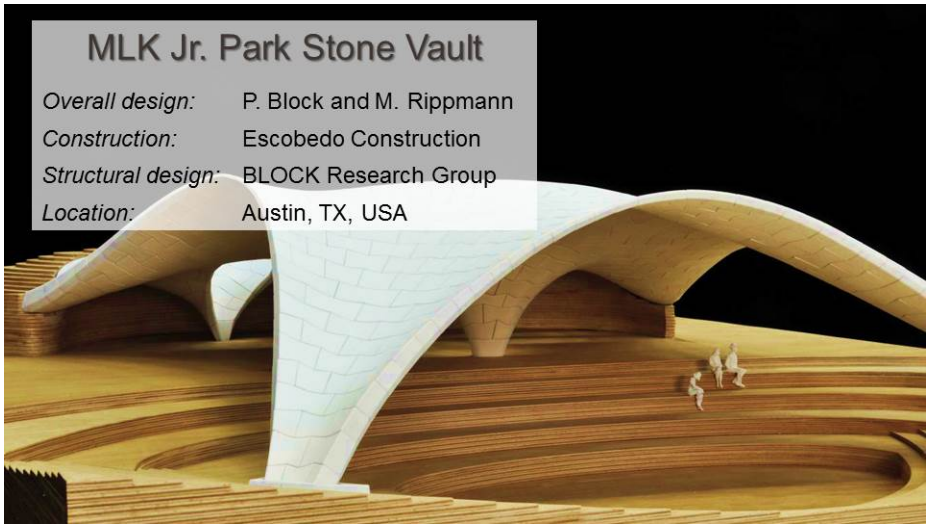


Figure 2.2: This masonry structure designed by P. Block is going to be constructed in Austin with local Texas limestone. It is self-supporting, as it can support its own weight with compression force only without reinforcement.

This system should include the basic geometric constraints such as planarity of faces. Besides simple geometric constraints, it should also be able to handle certain physical considerations, such as static equilibrium, or self-supporting. Both of geometric and physical constraints should be satisfied simultaneously at interactive rates. Further more, the system should be flexible enough to accommodate more considerations, such as matching given connectivity with user prescribed boundaries, and provide the possibility to create structures that have not been demonstrated before, such as non-height field self-supporting structures.

Solving those challenges is the main focus of this chapter. The core technical contribution is a general approach based on the proposed algorithm *Guided Projection*. This algorithm is applied to interactively create and edit structures satisfying both geometric and physical constraints, e.g., polyhedral meshes in static equilibrium. Most of the content presented here has been presented in [TSG+14].

To introduce *guided projection* as a general approach for computational design with

nonlinear constraints, a few basic geometric and physical considerations are discussed first in this chapter. Some of the capabilities are foundations for further extensions which are demonstrated in the following chapters with enriched applications.

For example, the planarity condition is extended to developability constraint in Chapter 3, which shares essentially the same underlying computational foundation. In parallel with the extension of hard constraints like planarity, soft energies such as fairness, which are discussed in greater details in the following paragraphs of the chapter, is extended to more general definitions of regularities based on local symmetries.

2.2 Prior Work

There are two seemingly independent roots to trace back related previous studies. On the one hand, design with geometric constraints in the modern freeform architectural context has been studied in [LPW+06, LXW+11, YYPM11, ZTY+12, BSWP12, POG13, DBD+13]. The work of [BSWP12] was even extended for animation in [BML+14] following a similar framework. On the other hand, design with physical constraints, especially static equilibrium, is a task for [Blo09, PBSH13, dGAOD13, LPS+13]. They study the modeling of statically sound shapes, especially self-supporting structures which are in static equilibrium with compressive forces only.

2.2.1 Constrained geometric designs

Polyhedral meshes are meshes with planar faces. Creating polyhedral meshes, especially planar quadrilateral (PQ) meshes, is considered one of the most fundamental tasks in architectural geometry, as planar panels are the most cost-effective elements available for surface paneling.

The work of Y. Liu et al. [LPW+06] is an earlier project in architectural geometry, which takes a combined approach with subdivision and planarity optimization to create planar-quadrilateral meshes. During the optimization, ensuring each face to be planar is done by requiring the sum of inner angles of each quadrilateral to be 2π, which is highly nonlinear. Later on, Y. Yang et al. [YYPM11] develop the idea of exploration of shape spaces to find variations of polyhedral meshes. The planarity condition is replaced by a cubic constraint by keeping the two diagonals of a quad touching each other.

A more recent work of R. Poranne et al. [POG13] introduces plane coordinates and formulates the planarity condition as the requirement that face vertices should satisfy the corresponding plane equations. With seemingly redundant variables and increased number of constraints, the reduced nonlinearity improves the computational efficiency as confirmed by the convergence rate. However, a staggered optimization scheme utilized and decoupled treatments of variables at different stages sacrifice the potential interactivity and system extensibility unfortunately.

Formulating constraints to be quadratic to enjoy the least amount of nonlinearity is a well-known trick in numerical optimization and scientific computing, see e.g.[BPC+10]. The efficiency of the quadratic formulation is further confirmed in this chapter. Formulating constraints to be at most quadratic serves as a rule of thumb for all the constraints. Those linear and quadratic constraints are incorporated in a flexible algorithm *Guided Projection*. The generality of this framework makes it suitable for the purpose of interactive design, and extendible to handle other desired constraints such static equilibrium.

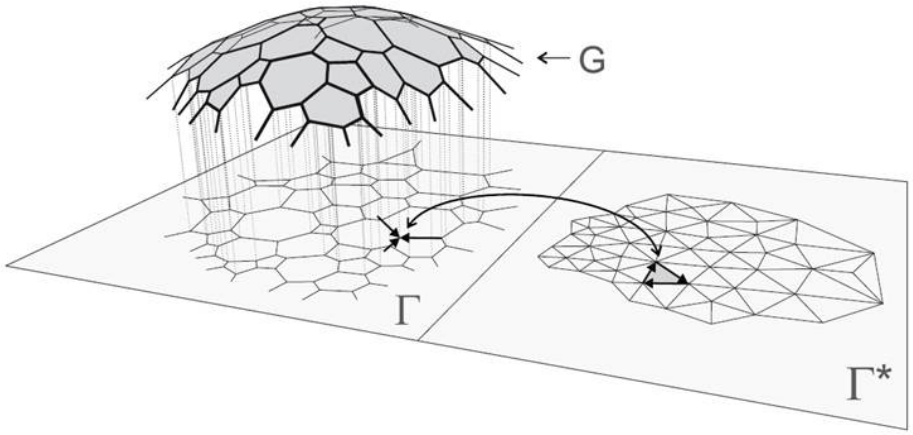


Figure 2.3: Thrust Network Analysis and Reciprocal Diagram [BO07]: a structure is in static equilibrium if there exists a thrust network within the structure, representing balanced internal forces. A classical and elegant way to ensure the existence of a balanced thrust network, and thus the horizontal balance of the structure, is through a valid reciprocal diagram of the top-view.

2.2.2 Static-aware design

In parallel with geometry, statics is a classical yet fast-developing field in the context of computational design. Physical models have been created by architects to find statically sound structures for more than a century, which was the origin of the concept of *form-finding*. More recently, computation has further accelerated this process by bringing the physical experiments to digital interactions.

A structure is in static equilibrium if there exists a valid *thrust network* within it, which represents balanced internal forces. An elegant approach to ensure the existence of a thrust network is through the help of top-views and reciprocal diagrams, which is an approach that can be dated back to Maxwell [Max64]. For local horizontal force balance at a vertex, the balanced forces form a closed polygon in the force diagram, or the reciprocal. If the forces are balanced everywhere, there exists a reciprocal diagram globally, and vice versa. Figure 2.3 presents an example. Such a top-view

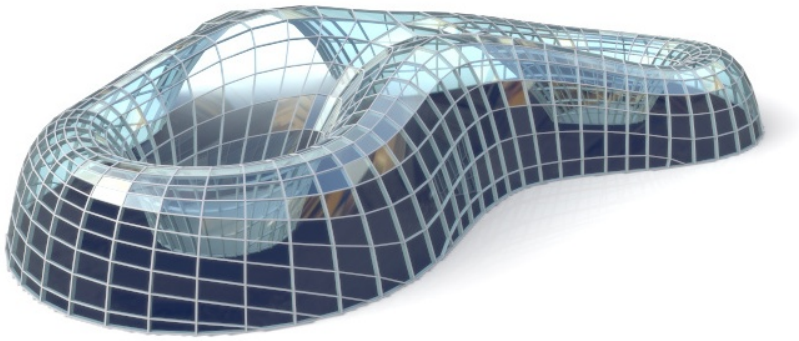


Figure 2.4: Vouga et al. [VHWP12] create self-supporting PQ meshes based on remeshing along relative principal curvature directions, which are determined by the reference shape and the associated Airy stress potential surface. As the connectivity is determined by the shape, there is no natural way to achieve interactivity as remeshing is a detached procedure from shape modification. Also due to the lack of user controllability over the mesh connectivity, boundary alignment cannot be easily achieved, leading to the unnatural trimmed boundary shown in this image.

and reciprocal based method is very powerful when the structure is a height field, but could be challenged otherwise.

2.2.3 Combining geometry with statics

There are few attempts at combining both the geometric considerations with the static-aware designs. Among all of the previous studies, the closest work along this direction is done by Vouga et al. [VHWP12], where self-supporting planar quadrilateral meshes are also generated. They present an approach in which static equilibrium and geometric considerations are treated at two independent stages. First, a self-supporting triangular mesh is created without the consideration of face planarity constraints. Next, a planar quadrilateral mesh is constructed by remeshing along relative principal curvature line directions given by the reference triangle mesh with respect to the associated Airy stress potential surface. Once a triangle mesh in static equilibrium is created, the connectivity of the quadrilateral mesh is roughly deter-

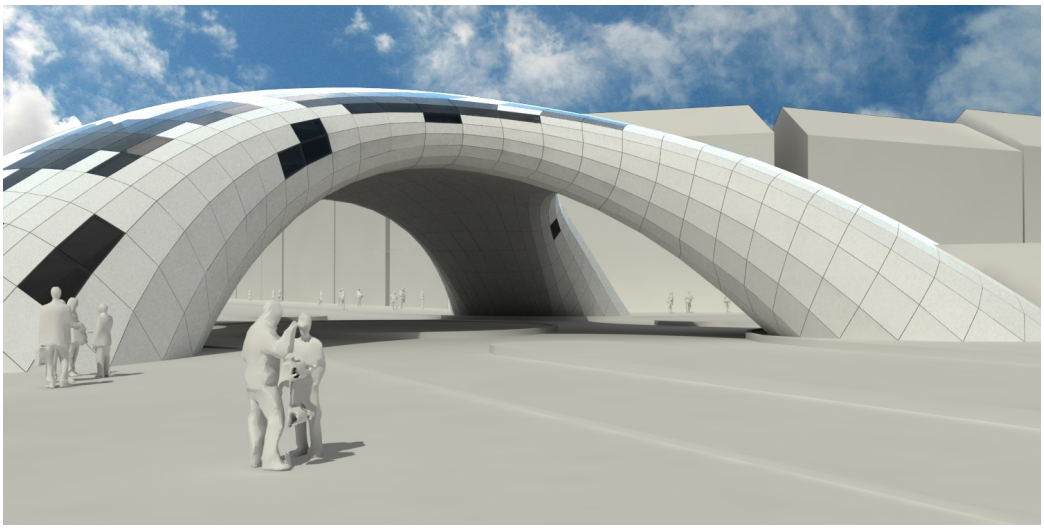


Figure 2.5: A self-supporting polyhedral mesh that is not a height field interactively modeled and created with the proposed approach. This structure is self-supporting, i.e., it can support its own weight with compression force only. It is also a polyhedral mesh, where every face is planar. Please note that the boundaries are aligned on the ground, matching the given boundaries with user specified connectivity of well-shaped quads and triangles only.

mined and cannot be further adjusted. Further, matching a given boundary with user prescribed connectivity is a notoriously difficult problem in engineering called boundary alignment, as shown in Figure 2.4. This task is getting even more challenging where the connectivity of the mesh cannot be controlled. Therefore, when static equilibrium and geometric constraints are required at the same time, the isolation of the two stages sacrifices the controllability of mesh connectivity, boundary alignment, as well as the interactivity of shape modeling.

2.3 Contributions and Overview

The following summarizes the functionalities of the proposed approach in the context of form-finding with polyhedral meshes. Those capabilities are roughly categorized to hard constraints like planarity and force equilibrium, soft energies taking care of

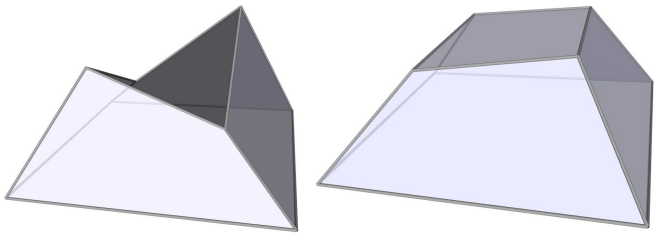


Figure 2.6: A basic goal of the proposed approach is to relocate the vertices of a mesh locally so that the faces are planar, while the modified shape is close to the original. Planarity serves as a basic constraint extendible to further geometric requirements such as developability, illustrated in Chapter3.

aesthetic or performance related considerations such as smoothness of the shape, as well as interactive interfaces for users to freely modify and explore possible designs within the constraint space. An example of the results is shown in Figure 2.5.

2.3.1 Geometric and physical constraints

Face planarization

A fundamental task of the proposed system, which is also a benchmark for shape modeling with constraints, is to relocate the vertices of meshes so that every face is planar, as shown in Figure 2.6. The planarity of faces is one of the most basic geometric constraints, maintained throughout the computation of our system ubiquitously. Its importance lies in both the applicable side as well as the algorithmic side. In architectural construction of panelization, planar panels are the cheapest options available, especially for the panels made from glass, wood and marbles. In computation, planarity could be enhanced together with offset properties, conic meshes or circular meshes. In the next chapter, we shall also see how planarity could be applied to further ensure developability of surfaces.

Offset related properties

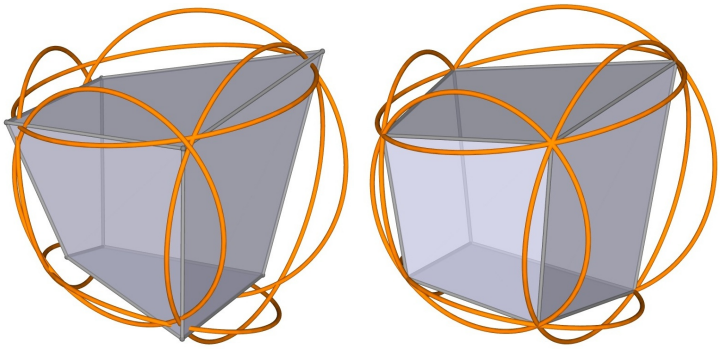


Figure 2.7: Special types of polyhedral meshes have constant offset polyhedral meshes. This property is especially important in architectural application for multi-layer constructions. Circular meshes, i.e., PQ meshes with faces inscribed to circles is important in this regard.

For the purpose of thermal insulation and structural soundness, many facades need to be constructed with multi-layer structures. For the computational design of those solutions, there is usually a further need to design two or more polyhedral meshes with parallel faces. Whilst layers of meshes could be all integrated into the computation, there is a more elegant way based on discrete differential geometry. It would not be difficult to derive that special types of polyhedral meshes, especially PQ meshes, enjoy the property of possessing constant face or vertex offsets. With these properties available, the computation could be done solely on a single layer of *principal meshes*, circular meshes, or conic meshes.

As shown in Figure 2.7, circular meshes are PQ meshes where every face is inscribed to a circum-circle, and conic meshes are those where every vertex star defines a right cone tangential to all the neighboring faces of that vertex. Both the circularity condition and conic condition could be incorporated into the proposed system in a uniform manner.

Boundary alignment

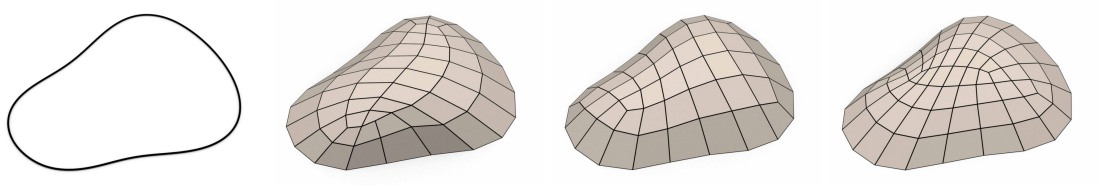


Figure 2.8: By incorporating given curves as boundary conditions, the task of boundary alignment, i.e., matching a given curve with desired mesh connectivity, could be achieved together with planarity, among other constraints. This has been considered a very challenging task from the perspective of engineering.

As illustrated in Figure 2.8, in architectural freeform design, a challenging task is to match a polyhedral mesh with desired connectivity with a given boundary curve, usually required by the limitation of the construction site, or the aesthetic considerations of the architect. By incorporating vertex-curve gliding conditions, boundary alignment could be incorporated during the computation as a common condition similar to other constraints in an efficient manner.

Static equilibrium

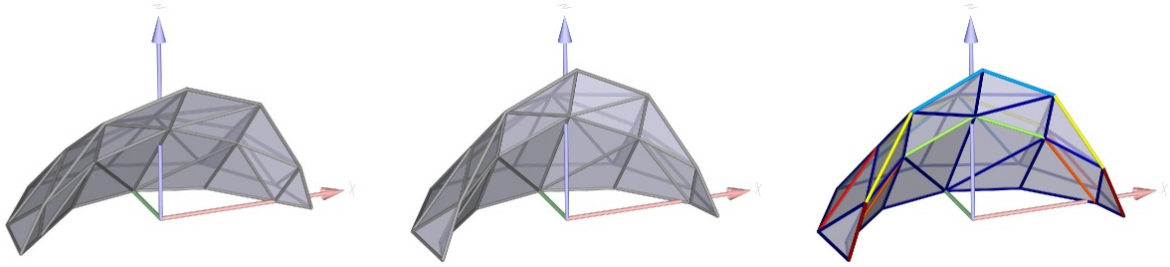


Figure 2.9: A basic physical constraint the system could handle is static equilibrium. From an input model (left), by relocating the vertices (middle) and finding the proper axial forces on the beams, a balanced thrust network is obtained (right).

As the basic mathematic formulations share great similarities, the proposed ap-

proach is naturally extended to handling physical constraints, and in particular, static equilibrium, as illustrated in Figure 2.9. By searching for the balancing forces and adjusting the proper locations of nodes, thrust networks could be brought to a balanced state. Beyond reproducing similar results as the existing approach, a novel capability of the proposed system is the capability to design self-supporting structures that are not height fields, namely, possessing overhanging parts that folds back to itself.

Computation with both geometric and static constraints

With proper adjustment of magnitude, multiple constraints can be taken care of simultaneously. As one of the most interesting examples shown in this chapter, combining the geometric considerations such as planarity of faces, together with physical constraints, e.g., self-supporting properties, creates polyhedral meshes in balance. Naturally, the novel non-height field self-supporting structure possessing overhanging part can also be realized with planar quads as well.

Other constraints

Incorporating target vertex locations as further conditions achieves handle-driven deformation within the unified framework, which further provides a user an intuitive tool to explore different solutions within the constraint manifold.

The constraints mentioned above are commonly required, but they are also only a small subset of all the constraints the proposed system could incorporate. Multiple other quantities could be easily formulated as well, such as the lengths of edges, the areas of faces. Besides equality constraints, inequality constraints could be converted to equalities with auxiliary variables, as shown in the following discussions. Further, global quantities such as the enclosed volume can also be encoded as constraints.

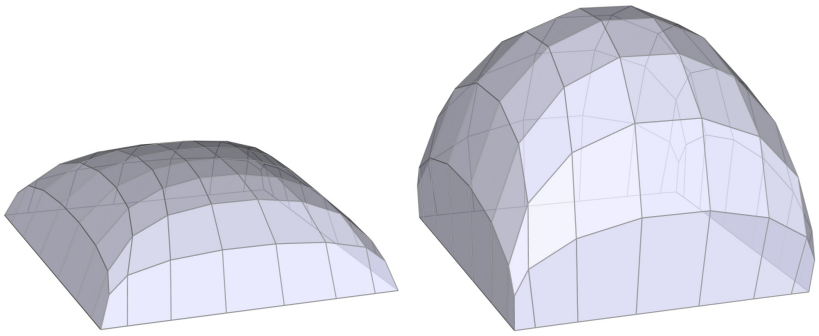


Figure 2.10: Global quantities such as the total length of beams, the sum of areas of panels as well as enclosed volumes, shown in this example, could be incorporated in the proposed approach.

For example, in Figure 2.10, the total volume of a polyhedral mesh with a square base is controlled. With some over simplified and reasonable assumptions, the cost of construction can also be measured by geometric quantities, and respected during the computation of the proposed system as well.

2.3.2 Soft energies

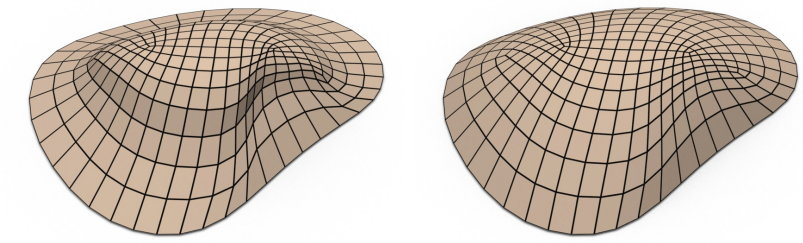


Figure 2.11: Simple aesthetic considerations, such as fairness, the discrete notion of smoothness could be utilized to find a polyhedral mesh with better appearances (right), among many solutions which satisfy solely the planarity constraints, such as the one shown on the left.

Among the solutions that satisfy the hard constraints, usually given by considerations of fabrications, there is still enough degrees of freedom to be taken advantage of to find solutions that perform better in certain criteria, for example, the overall

appearance or smoothness, as demonstrated in Figure 2.11. By encoding those considerations as soft energies, they could give the computation towards a result which not only satisfies the requirements of the desired hard constraints, but at the same time achieves the desired properties.

2.3.3 User interactions

Subdivision based multi-resolutional approach

In conjunction with a typical graphics application, the proposed approach could be combined with subdivision. As shown in Figure 2.12, a user could control the coarse control polygon, and the computation is applied consistently to a finely subdivided result. Each time, the computation starts from the unconstrained subdivided mesh, but as the computation is guided by consistent choices of soft energies, the intended modifications on the control structure are reflected on the final results after computation. Another example is shown in Figure 2.14.

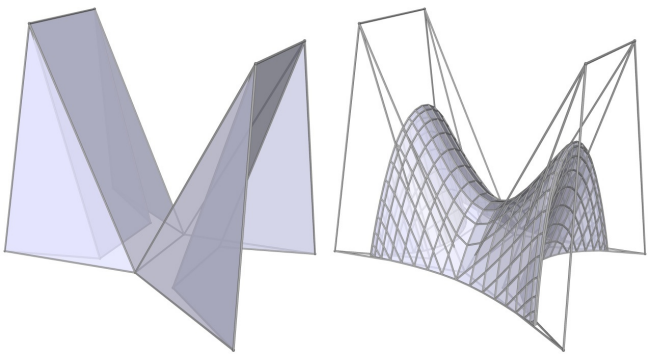


Figure 2.12: Control structures and subdivided mesh for the consistent re-initialization of final result. Please note that the boundaries are subdivided in a way similar to B-spline curves, so they do not shrink.

Handle driven manipulation

Another graphics deformation tool is handle-driven manipulation. This function commonly exists in commercially available shape modelers. It allows a user manipulate the shape of a model by dragging selected handles of the mesh, usually vertices, as shown in Figure 2.13. When the handles are relocated by the user, the displacement is intuitively propagated to a nearby region for smooth and intuitive deformation. With the proposed approach, target vertex handle location or face orientation could be regarded as extra hard constraints to be incorporated to the system and handled in a similar manner. As the other hard constraints are kept together with the encoded handle hints, the deformed meshes also satisfy the desired constraints.

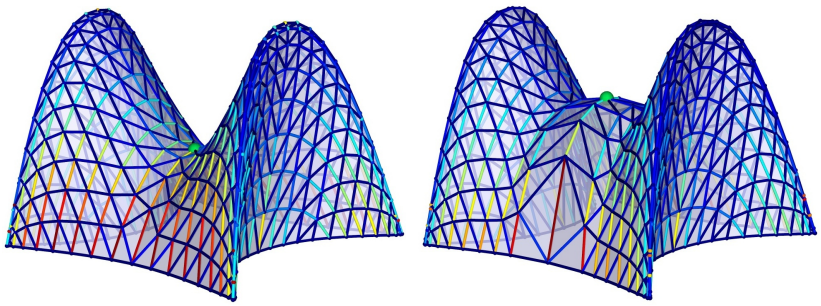


Figure 2.13: Incorporating target vertex locations as additional conditions achieves handle-driven deformation within the unified framework. It further provides a user an intuitive tool to explore different solutions within the constraint manifold.

2.4 Algorithms and Results

2.4.1 Iterative algorithm: Guided Projection

The proposed algorithm, *Guided Projection*, is intended to be the foundation for the interactive design. The problem could be formulated as solving a collection of hard constraints, including geometric ones such as planarity, physical ones such as

force equilibrium, as well as other customized constraints provided by the user. As the collections of hard constraints are usually under-determined, there are generally enough degrees of freedom to find solutions that satisfy certain soft targets better, such as fairness, or smoothness of the designed surfaces. Most importantly, all of the considerations should be achieved in interactive rates on an ordinary platform.

The key ideas are quite simple. First, by introducing auxiliary variables, the constraints are at most quadratic, and each equation only involves a few variables. Second, the system of constraints are solved together with fairness energies as regularizers. However, instead of solving a constrained optimization problem, the soft energies such as fairness terms are treated as guidelines during the computation, and not optimization goals. In other words, the goal is not to find the minimizer of certain energies globally, but rather a reasonably good one in a solution space. Therefore, the algorithm is named as Guided Projection: projecting onto the design space guided by soft energies.

Guided Projection usually starts with a set of initial variables as well as a system of quadratic equations, usually under-determined and inconsistent due to inter-dependencies, as redundant constraints are allowed.

$$\psi_i(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T H_i \mathbf{x} + \mathbf{b}_i^T \mathbf{x} + c_i = 0,\ i = 1, \ldots, N, \tag{2.1}$$

for an iteration starting from $\mathbf{x}^k$, we linearize the set of quadratic equations by taking the first order Taylor expansions, or geometrically, approximating each quadratic surface by a hyper-plane.

$$\psi_i(\mathbf{x}) \approx \psi_i(\mathbf{x}^k) + \nabla\psi_i(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k),\ i = 1, \ldots, N. \tag{2.2}$$

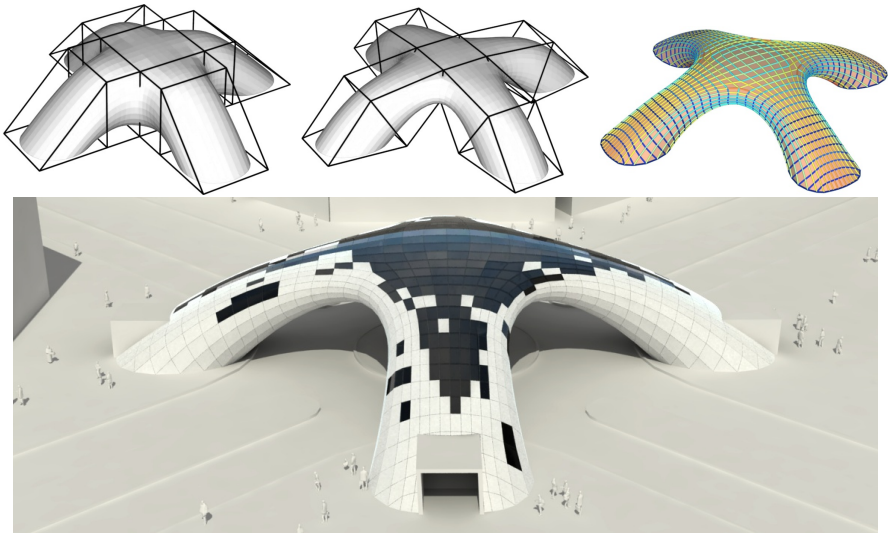


Figure 2.14: Starting with a control structure and an initial subdivided surface (top-left), user manipulation of the control structure which leads deformed initial structure after direct subdivision. When variables are properly initialized based on the subdivided mesh, e.g., force densities are properly initialized, the iterative procedure of *Guided Projection* brings the structure to a polyhedral mesh in force balance, matching the boundaries prescribed.

Written in the matrix form, it is:

$$H\mathbf{x} = \mathbf{r}, \quad H = \begin{bmatrix} \nabla\varphi_1(\mathbf{x}_n)^T \\ \vdots \\ \nabla\varphi_N(\mathbf{x}_n)^T \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} -\varphi_1(\mathbf{x}_n) + \nabla\varphi_1(\mathbf{x}_n)^T\mathbf{x}_n \\ \vdots \\ -\varphi_N(\mathbf{x}_n) + \nabla\varphi_N(\mathbf{x}_n)^T\mathbf{x}_n \end{bmatrix}. \tag{2.3}$$

The linearized constraints are solved together with linear equations encoding soft energies. Those soft energy terms reflect the desired but not mandatory considerations like fairness. While the linearized constraints are given constant high weights, the coefficients for the soft energy terms are reduced until vanishing through out the computation to ensure the hard constraints are truly respected.

For two reasons, a small proximity term is added: first, the system of linear equations may be under-determined, adding this term prevents the rank deficiency;

second, this term prevents the solution drifting away from the previous iterations.

$$\|H\mathbf{x} - \mathbf{r}\|^2 + \|\varepsilon(K\mathbf{x} - \mathbf{s})\|^2 + \|\delta(\mathbf{x} - \mathbf{x}^k)\|^2 \to \min . \tag{2.4}$$

This procedure is repeated with less emphasis on soft energies, and eventually solving a system with only hard constraints and the proximity term, to ensure that the hard constraints are satisfied to a high precision.

2.4.2 Variables and hard constraints

Following the introduction of the capabilities of the proposed approach, details on the formulations of the hard constraints as well as the energy terms are given below. As the variables kept during the computation inherently coupled together with the hard constraints, due to the fact that redundant variables are used to keep the constraints at most quadratic, therefore, they are introduced together in the section. For example, while the shape of a mesh $\mathcal{M} = (V, E, F)$ could be retrieved by the information on the connectivities together with the vertex locations $\mathbf{v}_i, i = 1, \ldots, \|V\|$ as $3\|V\|$ variables, we add extra face normals to facilitate the planarity requirement of each face with extra $3\|F\|$ variables. More auxiliary are introduced when the related hard constraints are incorporated into the system of equations to be solved.

Planarity

For face planarity, we introduce a face normal $\mathbf{n}_f$ for every face. As shown in Figure 2.15, the planarity of each quad is guaranteed by the orthogonality of the face normal and its adjacent face edges:

$$(\mathbf{v}_i - \mathbf{v}_{i-1}) \cdot \mathbf{n}_f = 0. \tag{2.5}$$

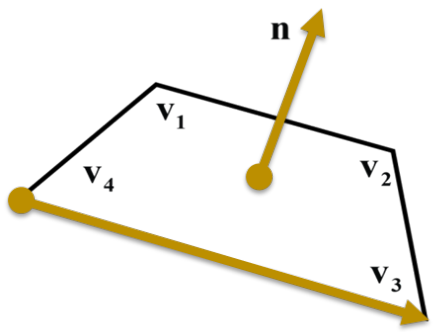


Figure 2.15: Face normals are used to ensure planarity by its orthogonality to each face edge. They are treated as variables during the computation, updated at the same time as the vertices.

The orthogonality conditions are required on all the face edges. Such seemingly redundancy helps us to treat all the variables equally, and to make the solver more symmetric with respect to each vertex. Here, we allow redundant constraints, and require such orthogonality relations for all the face edges explicitly. In fact, we can leave out one of the edges, as it would automatically satisfy the orthogonality condition if others are all satisfied. However, we still require all the orthogonality constraints here, as such redundancy can ensure all the vertices are equally treated.

Circularity

As mentioned in the overview, possessing offset property is favored for the construction of multi-layer structures. Two typical polyhedral meshes with constant offsets are circular meshes and conic meshes. For circular meshes, every face needs to be inscribed to a circum-circle. Thus, for this circularity condition, we further introduce the center of each circle as shown in Figure 2.16, and require the face vertices to keep the same distance from the circle center:

$$\|\mathbf{v}_i - \mathbf{c}\|^2 = \|\mathbf{v}_j - \mathbf{c}\|^2. \tag{2.6}$$

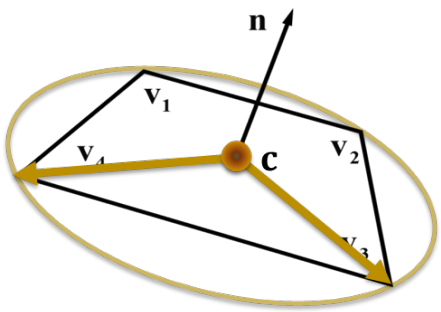


Figure 2.16: Circularity: on the top of planarity, the center of each circle is introduced, and every face vertex is required to keep the same distance from the center.

This condition is equivalent to requiring that the chord is orthogonal with respect to the apothem, the connecting line between the center of the circle and the mid-point of the chord:

$$\left(\frac{\mathbf{v}_i + \mathbf{v}_j}{2} - \mathbf{c}\right) \cdot (\mathbf{v}_i + \mathbf{v}_j) = 0. \tag{2.7}$$

With similar consideration, conic condition could be formulated as quadratic equations as well.

Edge lengths, face areas, and enclosed volumes

The lengths of edges as well as the areas of faces can also be easily encoded as quadratic equations. Incorporating those quantities could be further combined to the considerations of cost, as well as other measurements such as the total enclosed volumes.

For every edge connecting two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$, the edge length l_{ij} is simply

$$l_{ij}^2 = (\mathbf{v}_i - \mathbf{v}_j) \cdot (\mathbf{v}_i - \mathbf{v}_j). \tag{2.8}$$

For a face f_k with consistently ordered face vertices $(\mathbf{v}_k^1, \mathbf{v}_k^2, \ldots, \mathbf{v}_k^{|f_k|})$, the vectorial

area $\mathbf{a}_k$ is:

$$\mathbf{a}_k = \frac{1}{2} \sum_{i=1}^{|f_k|} \mathbf{v}_k^i \times \mathbf{v}_k^{i+1} \tag{2.9}$$

where the index in the sum is taken in modulo $|f_k|$. The associated scalar area is simply the norm of this vectorial quantity:

$$a_k^2 = \mathbf{a}_k \cdot \mathbf{a}_k. \tag{2.10}$$

With the help of vectorial areas, the enclosed volume V of a mesh, defined over the horizontal x, y-plane, is simply the sum of the product of the z-component of the vectorial area times the average height of the column:

$$V = \sum_{k=1}^{|F|} (\mathbf{a}_k \cdot \mathbf{e}_3) \left(\frac{1}{|f_k|} \sum_{i=1}^{|f_k|} \mathbf{v}_k^i \cdot \mathbf{e}_3 \right). \tag{2.11}$$

With a proper orientation and placement of the mesh, a downward oriented face over the horizontal plane contributes negatively to the total volume. Therefore, when there is any overlapping part along the vertical direction, the extra volume is canceled out, which renders this definition suitable for structures even with overhanging parts, as shown in Figure 2.17.

However, maintaining or controlling such global quantities involves too many variables in a single equation, which leads to significantly denser linear systems and much slower computational speeds.

Static equilibrium

As with geometric constraints, physical requirements can be similarly formulated. The force balance condition requires that the resultant force at each vertex balances

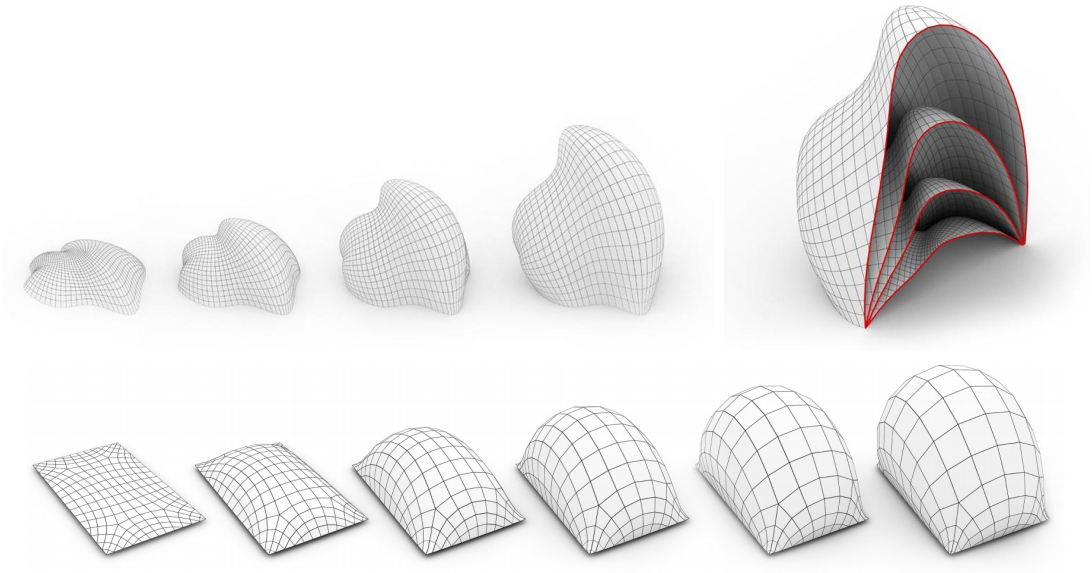


Figure 2.17: Total volume control: by increasing the total volume or area while maintaining face planarity and mesh fairness, polyhedral meshes could be inflated like bubbles.

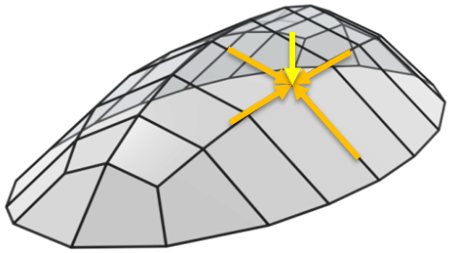


Figure 2.18: Force balance: At every vertex, the resultant force from adjacent beams should balance the local load, which are usually constituted of the weights of adjacent panels or adjacent beams.

the vertical load there, as shown in Figure 2.18. In our assumption, the loads are contributed from adjacent faces or beams. For each edge, we define a so called force density w_{ij}, force per unit length. Based on selected material and predetermined cross section, the line mass density of beams, defined as mass per unit length, are regarded as given constant ρ^e. Similarly, for faces with constant thickness, the mass of panels could be computed based on the area and the mass per unit area given as ρ^f. Figure 2.19 shows a result where the loads are proportional to panel areas. Then

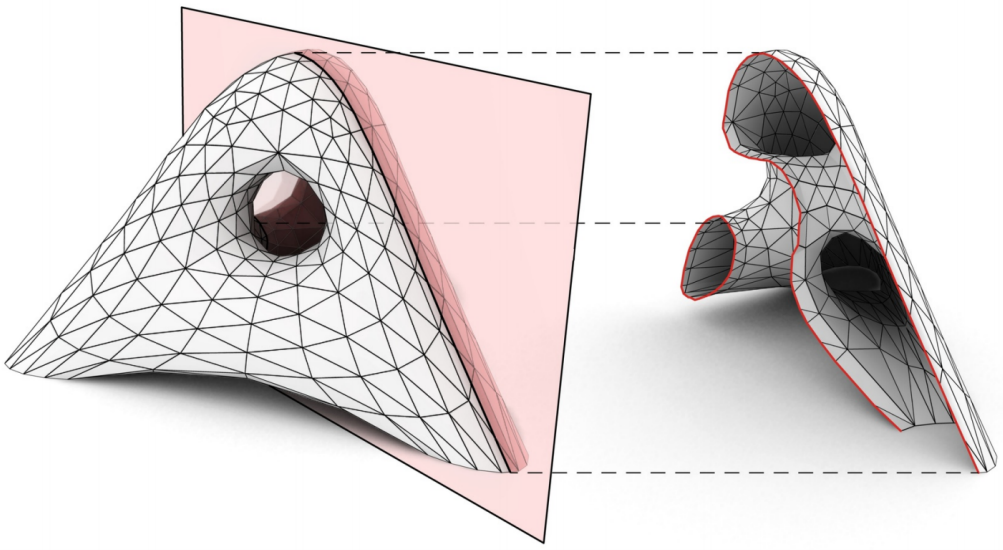


Figure 2.19: A non-height field self-supporting structure where the load on each vertex is assumed to be proportional to the summed areas of its neighboring faces.

the force balance condition is also quadratic. Similarly, face areas and edge lengths can be written as quadratic terms as well:

$$\sum_{j:\ \mathbf{v}_i\mathbf{v}_j\in E} w_{ij}(\mathbf{v}_j-\mathbf{v}_i) = \begin{pmatrix}0\\0\\1\end{pmatrix}\left(\sum_{j:\ \mathbf{v}_i\mathbf{v}_j\in E}\frac{1}{2}\rho^e l_{ij} + \sum_{k:\ \mathbf{v}_i\in f_k}\frac{\rho^f a_k}{|f_k|}\right). \tag{2.12}$$

Here, the edge lengths l_{ij} are kept as variables based on Equation 2.8, so are face areas a_k based on Equation 2.9 and Equation 2.10. Based on our assumptions, ρ^e and ρ^f are regarded as constants. However, it would not be difficult to keep them as variables as well.

Inequalities

For inequality conditions, we introduce auxiliary variables, and convert them to equality conditions. For example, the self-supporting condition allows only compression

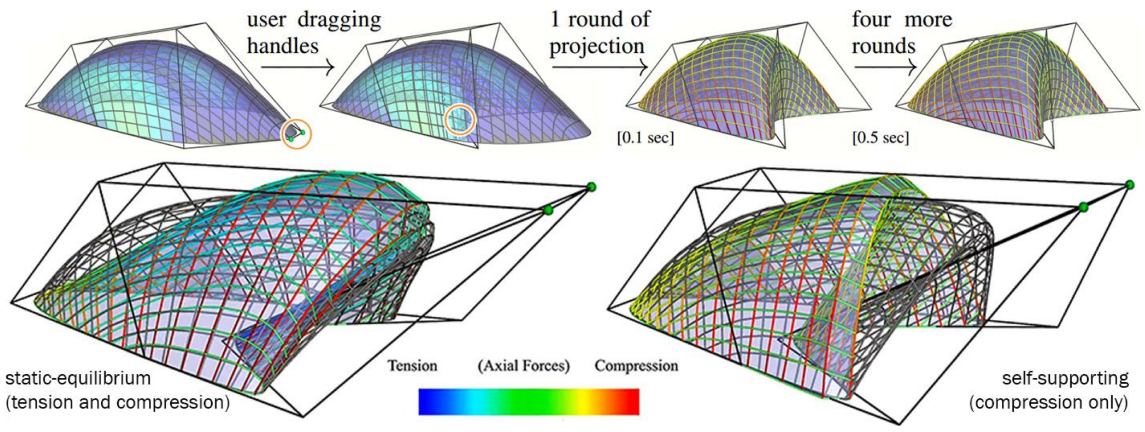


Figure 2.20: Screen shots of interactive design with planarity and statics based on subdivision modeling, coupled with guided projection onto the constraint manifold. Note that our method provides perfect user-controlled boundary interpolation: Partly the boundary is a mesh polyline, partly the boundary consists of diagonals. Please also note that in contrast to the mesh in static equilibrium with both tension and compression (bottom-left), self-supporting allows only compression, and prevents the top of the structure further leaning forward (bottom-right).

forces, and the force densities should all be nonnegative. To satisfy this, we simply require them to be the squares of the dummy variables. An effect of those inequality constraints are demonstrated in Figure 2.20.

2.4.3 Soft Energies

Among the solutions that satisfy the strictly required *hard* constraints, other desired properties may be further required, such as aesthetics. Though some of those considerations are quite subjective, some could be extracted and formulated easily, as introduced below.

Vertex fairness

A basic measurement of aesthetics would be the smoothness of the shape, or discretely, the *fairness*. For triangle meshes, a commonly used smoothness term is the Laplacian

smoothness, which requires each vertex to be close to the average of its neighboring vertices. For a quadrilateral mesh, there is a more natural way to define fairness based on the two families of polylines by ensuring every polyline is smooth discretely, as demonstrated in Figure 2.21.

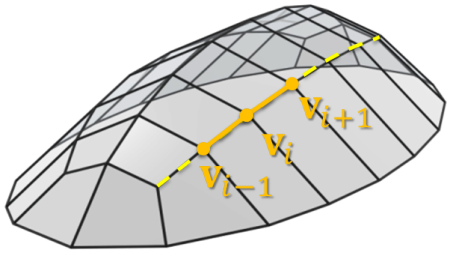


Figure 2.21: To ensure the modeled shape is discretely "smooth", fairness for quad meshes could be required so that each vertex is close to the mid-point of its two neighbors on every polyline.

For each vertex on a polyline, we require it to be close to the mid-point of its two neighbors. This requirement is applicable for all valence-4 vertices, as they are regular crossing of two polylines. For a vertex $\mathbf{v}_i$ with four neighboring vertices $\hat{\mathbf{v}}_i^k, k = 1, \ldots, 4$, this condition reads:

$$\mathbf{v}_i \approx \frac{1}{2}\left(\hat{\mathbf{v}}_i^1 + \hat{\mathbf{v}}_i^3\right) \approx \frac{1}{2}\left(\hat{\mathbf{v}}_i^2 + \hat{\mathbf{v}}_i^4\right) \tag{2.13}$$

Vertices with valences other than 4 may not be regarded as regular crossings of two polylines, they are required to be close to the centroid of their one ring vertex neighborhoods, which is similar to the smoothing based on the graph Laplacian operator:

$$\mathbf{v}_i \approx \frac{1}{n} \sum_{j:\ \mathbf{v}_i\mathbf{v}_j \in E} \mathbf{v}_j. \tag{2.14}$$

As demonstrated in Figure 2.22, fairness has a strong effects in controlling the

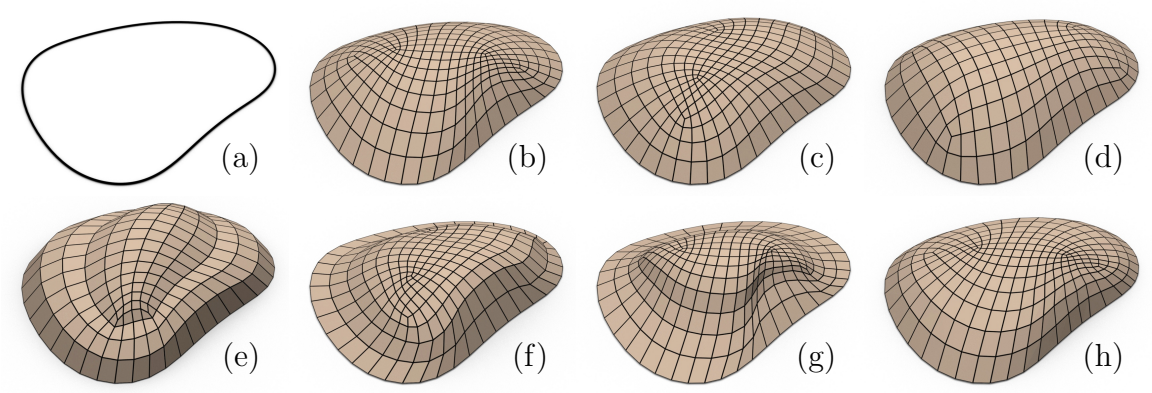


Figure 2.22: Exploring the design space and experiencing its limitations: Firstly we observe that polyhedral meshes filling the same boundary (a) and having different combinatorics, generally lead to similar shapes, see (b), (c), (d). User interaction may yield meshes which either have sharp features, such as (e), or are skewed, such as (f), or both (g). Subsequent optimization for equilibrium forces acts like smoothing and reconstructs a dome-like shape apparently largely independent of the combinatorics, see (h).

feasible shapes, even across different connectivities. This provides advantages that the polyhedral meshes created are less sensitive to the choice of combinatorics. At the same time, this also restricts the design space to a limited range, which prevents the idea of shape space exploration to be performed in very meaningful ways.

Vertex tangential fairness

As vertex fairness straightens every polyline to possess equally distributed vertices, it introduces the side effects of mesh flattening. As shown in Figure 2.23, to prevent such over-smoothing, fairness could be applied only along the tangential direction as *vertex tangential fairness*:

$$\begin{bmatrix} \mathbf{t}_{i,1}^T \\ \mathbf{t}_{i,2}^T \end{bmatrix} \mathbf{v}_i \approx \frac{1}{2} \begin{bmatrix} \mathbf{t}_{i,1}^T \\ \mathbf{t}_{i,2}^T \end{bmatrix} \left(\hat{\mathbf{v}}_i^1 + \hat{\mathbf{v}}_i^3 \right) \approx \frac{1}{2} \begin{bmatrix} \mathbf{t}_{i,1}^T \\ \mathbf{t}_{i,2}^T \end{bmatrix} \left(\hat{\mathbf{v}}_i^2 + \hat{\mathbf{v}}_i^4 \right). \tag{2.15}$$

Here, $\mathbf{t}_{i,1}^T$ and $\mathbf{t}_{i,2}^T$ are two orthogonal unit tangent vectors computed from the approximated vertex normal of $\mathbf{v}_i$ at the beginning of each iteration. As only the

geodesic curvature is minimized, while the normal curvature is left intact, this energy is feature preserving. Similar to vertex fairness energy, to apply tangential vertex fairness energy to vertices with valences other than 4, it requires that:

$$\mathbf{v}_i \approx \frac{1}{n} \begin{bmatrix} \mathbf{t}_{i,1}^T \\ \mathbf{t}_{i,2}^T \end{bmatrix} \sum_{j:\ \mathbf{v}_i\mathbf{v}_j \in E} \mathbf{v}_j. \tag{2.16}$$

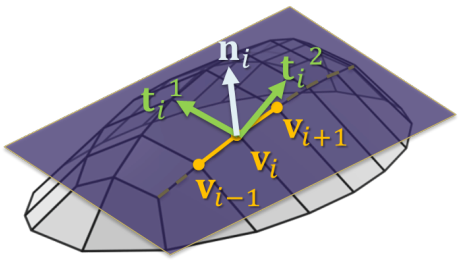


Figure 2.23: Tangential fairness only affects the geodesic portion of the discrete curvature of the polylines, which is feature preserving as it no longer have the tendency to flatten meshes as the unmodified vertex based fairness energy.

Face normal fairness

Similar to the fairness of vertices, the regularity of the face normals, or the Gauss images are also important measurements of the aesthetics of the meshes. The fairness energy of the face normals on the Gaussian sphere, or the closeness of neighboring face normals. For a general quadrilateral mesh, let n_k be the normal of a face, and $\hat{n}_k^i, i = 1, \ldots, 4$ be the normals of adjacent faces, then the tangential polyline fairness is applicable for those face normals:

$$\begin{bmatrix} \mathbf{t}_{i,1}^T \\ \mathbf{t}_{i,2}^T \end{bmatrix} \mathbf{n}_i \approx \frac{1}{2} \begin{bmatrix} \mathbf{t}_{i,1}^T \\ \mathbf{t}_{i,2}^T \end{bmatrix} \left(\hat{\mathbf{n}}_i^1 + \hat{\mathbf{n}}_i^3 \right) \approx \frac{1}{2} \begin{bmatrix} \mathbf{t}_{i,1}^T \\ \mathbf{t}_{i,2}^T \end{bmatrix} \left(\hat{\mathbf{n}}_i^2 + \hat{\mathbf{n}}_i^4 \right). \tag{2.17}$$

As the face normals are unit vectors, there is no need to differentiate tangential face normal fairness from face normal fairness.

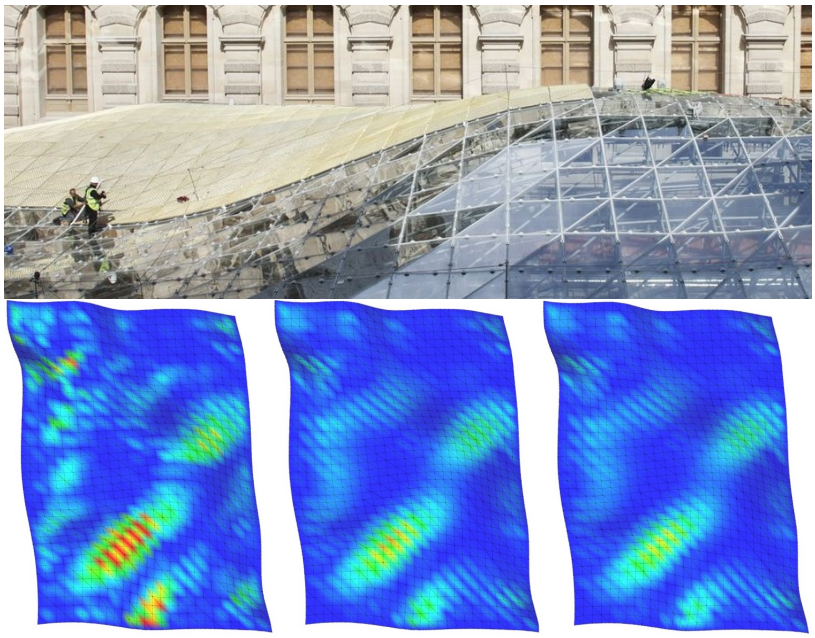


Figure 2.24: Top: the construction of the freeform roof for the Islamic Department in Louvre museum, Paris. The surface is rationalized as a hybrid mesh consisting planar quadrilaterals and triangles. Bottom: visualization of normal variations around vertices for the original mesh (left), and results after 2 and 5 iterations of face normal fairing are applied.

Figure 2.24 illustrates a typical usage of face normal fairing applied to a recent architectural design, the freeform roof of the Islamic Department of the Louvre museum resembling a flying carpet. The architectural design is rationalized as a hybrid mesh consisting both quads and triangles. For this original design, the vertices of the mesh have fixed horizontal coordinates according to a grid layout to simplify the computational task. The reduction of the degrees of freedom results in undesirable non-smooth appearances noticeable in the final construction. With the proposed approach, this restriction could be easily circumvented, as the vertex locations are fully adjustable when the normal fairness, face planarity and closeness to reference surfaces

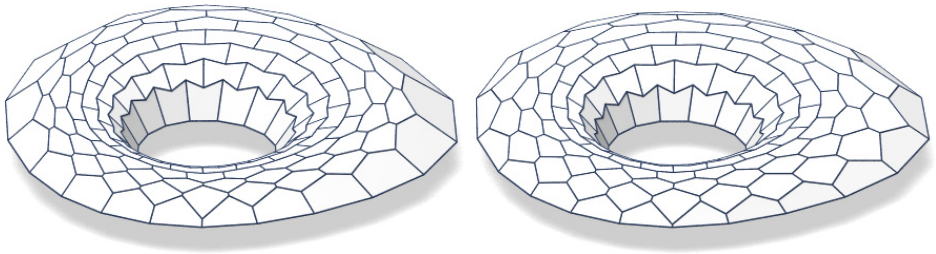


Figure 2.25: Equal face area as soft constraints. From left to right: a planar hexagon mesh has a better face area distribution while the planarity of each face is maintained. This is done by computing the average face area prior to each iteration, and then require each face to have a face area that is close to that average.

are aimed simultaneously. Starting with the original mesh (left), iterations of guided projection reduce the variations of face normals whilst keeping every quad planar.

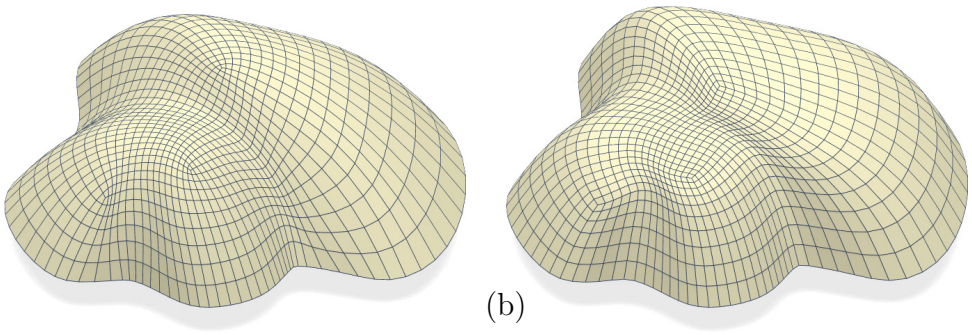
Equal face areas, equal edge lengths

In engineering, panels with similar sizes and beams with lengths fallen into certain ranges are desired. By pre-computing the average edge length of the mesh, or the average face area prior to each iteration, the distribution of edge lengths or face areas could become more even by requiring every face area or every edge length to be similar to the average quantity. Figure 2.25 shows such an example.

More soft energies

More soft energies could be devised according to specific needs. For example, in Figure 2.26, faces are encouraged to be closer to parallelograms. In Figure 2.27, the concept of fairness is extended for the axial forces, to optimize the structure towards a evener distribution of forces.

The soft energies presented above are not exhaustive. In the next chapters, a family of soft energies to regularize more general polyhedral surfaces are introduced



(a) (b)

Figure 2.26: By requiring each face to be closer to parallelograms, the effect of combinatorics on the shape is more prominent, and the features start to emerge.

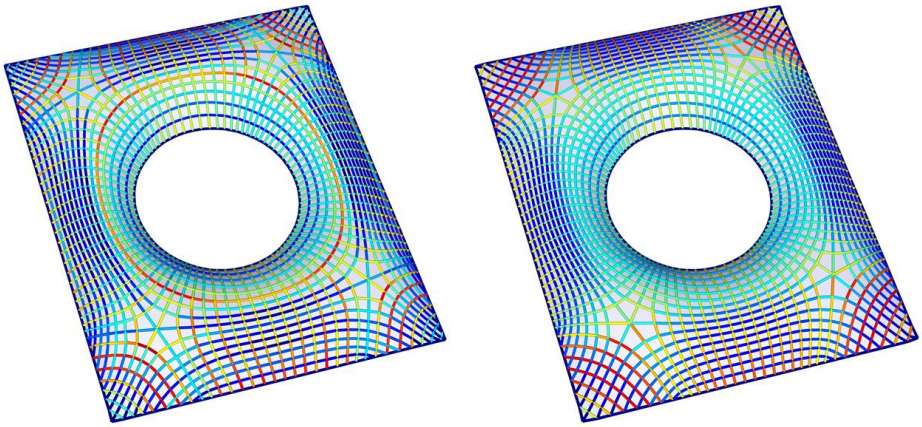


Figure 2.27: Force fairness encourages beams to have similar stresses. It improves the static performance of the structures as the maximal strains are reduced.

based on local symmetries.

2.4.4 Initialization

As shown in Figure 2.28, a basic task of the initialization is to have a reasonable starting point to run the guided projection algorithm. For the enumeration of initial connectivity, the work of Chi-Han is used. For static equilibrium, the forces are initialized in a least squares sense. For self-supporting structures, initial force densities are computed by the method of [Liu10], as a bounded least squares problem.

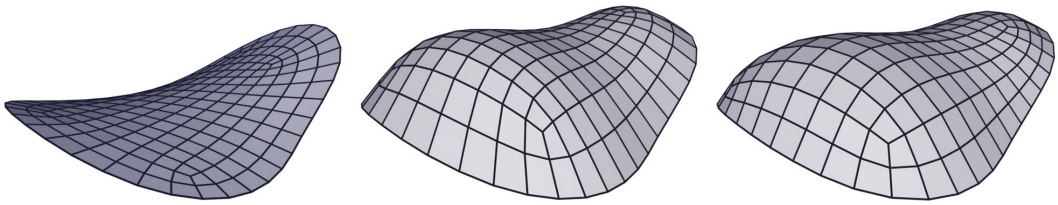


Figure 2.28: Initialization of a polyhedral mesh satisfying prescribed boundary conditions: starting with a smooth mesh without planar face which has its connectivity enumerated. The vertices are raised up while fairness, top-view closeness are required at the same time (middle), then further modifications could be used afterwards.

2.5 Discussion

2.5.1 Comparisons with previous methods

Thanks to the simplified algebraic structures based on quadratic formulation and a regularized Newton-like method, guided projection is highly efficient in computation. The advantage over state of the art methods [BSWP12] and [DBD$^+$13] are tested in the convergence analysis of planarity optimization shown in Figure 2.29 obvious . More comparisons of these two methods against previous methods could be found in [DBD$^+$13], which has already demonstrated great improvement over the methods before it.

With faster speed in computation, interactive manipulation could be achieved without requiring high cost hardware, and most of the examples demonstrated in this chapter could be interactively manipulated in a single core implementation of an ordinary laptop.

2.5.2 Implementation details

A prototypical system is implemented in C++ on Linux with QtCreator. The library QGLViewer is used for the visualization and user interactions, and the linear algebra

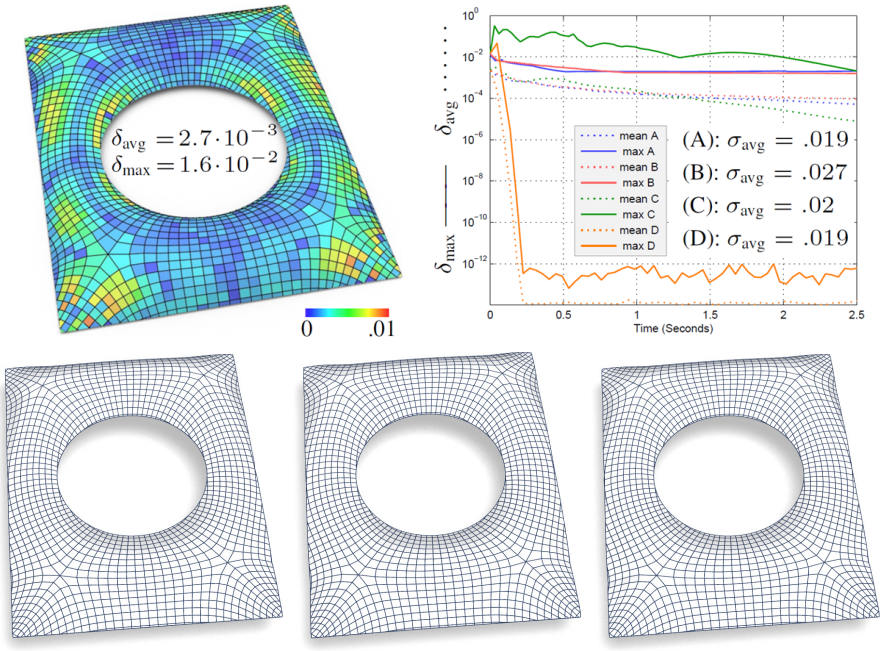


Figure 2.29: Convergence plot of planarity against computational time, it confirms the efficiency of Guided Projection. Here, the planarity is measured as distance of diagonals normalized by the average edge length per face. A: S. Bouaziz et al. [BSWP12], B: B. Deng et al. [DBD+13], C: Guided Projection with cubic constraints, D: Guided Projection with quadratic constraints.

manipulations is done with the help of the libraries Eigen and SuiteSparse. The half-edge mesh data structure is based on OpenMesh. Tests running with Intel® Core™ i5-3320M CPU [2.60GHz and 8G RAM] achieve interactive rates for typical examples shown in this thesis. More details of the timing could be found in [TSG+14].

2.6 Conclusion

This chapter introduces the algorithm *Guided Projection*, including the hard constraints, the soft energies, and the iterative procedure. Guided Projection could be applied to a variety of scenarios throughout the interactive design phase. As a basic and benchmark task, it could be used to planarize the faces of given meshes in a highly efficient manner. Besides geometric constraints, static equilibrium could also be achieved, independently or simultaneously, as another example shown in Figure 2.30.

Beyond local requirements, global quantities could be similarly constrained, such as total area or volume, as they can also be easily expressed through quadratic terms. Unfortunately, such expressions are usually dense and may make the computation slower. One possible future direction to look into is how to accelerate the computation by splitting the sparse terms from the dense ones, possibly through Alternating Direction Method of Multipliers (ADMM).

For general purposes, the efficiency of *Guided Projection* is demonstrated through examples including planarization of polyhedral meshes, as well as interactive design of meshes which are constrained by both geometric constraints and static equilibrium requirements. Based on the content presented in this chapter, in the next chapters, more extensions and related applications are further explored, including the design of developable surfaces and polyhedral patterns.

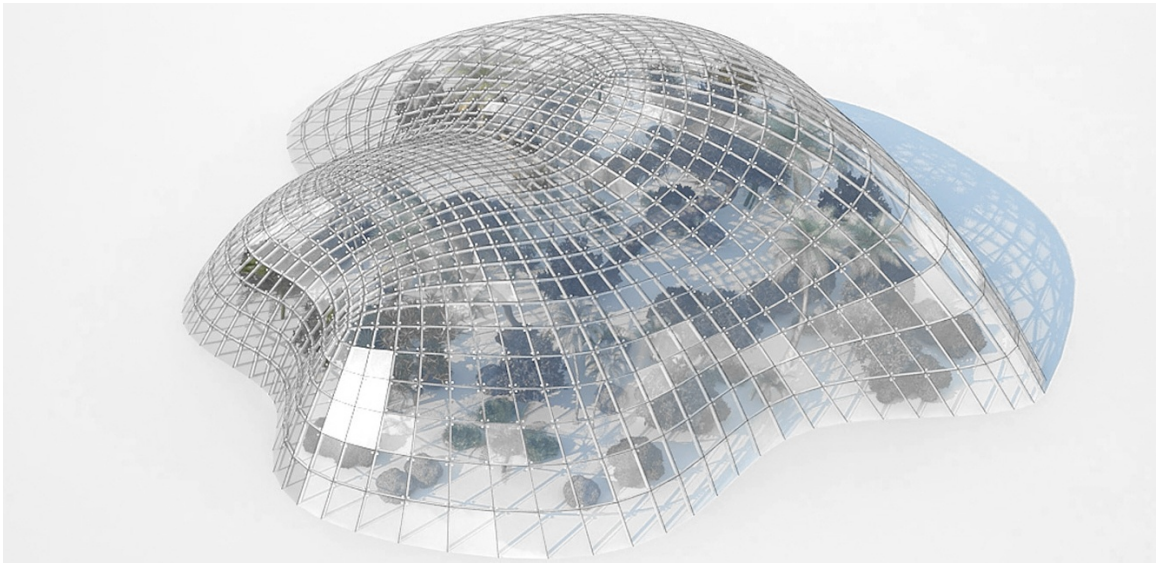


Figure 2.30: Another architectural freeform design "greenhouse" in force balance panelized by planar glass faces filling out a given boundary on the ground.

Chapter 3

Interactive Design of Developable Surfaces

3.1 Introduction

The previous chapter introduces *Guided Projection* as a basic framework of computational design with non-linear constraints. The constraints discussed previously include geometric ones like planarity of faces and physical ones such as static equilibrium. This chapter focuses on the extension of the geometric properties, from polyhedral meshes to developable surfaces, which are surfaces that could be flattened to a plane without tearing and stretching. The content presented here is based on [TBWP]. Figure 3.1 presents two examples with developable surfaces.

To extend polyhedral meshes to developable surfaces, the planarity constraints are inherited by and generalized to developability and several associated conditions. The proposed extension has significances for both the technical side as well as the practical side.

On the technical side, planarity of faces is a discrete concept by definition. On the other hand, the developability is a property defined on general continuous surface.

This implies that achieving this condition could be a problem with infinite dimensions for smooth surfaces.

A key contribution presented in this chapter is the formulation of the developability constraint. In addition, the associated computational approach is proposed based on *Guided Projection* introduced in the previous chapter. By combining the benefits of spline based representations and mesh based constraint formulations, the infinite dimensional problem is transformed to a finite one. With the proposed formulation, the constrained surface is provably developable everywhere up to numerical precision without suffering discretization errors.

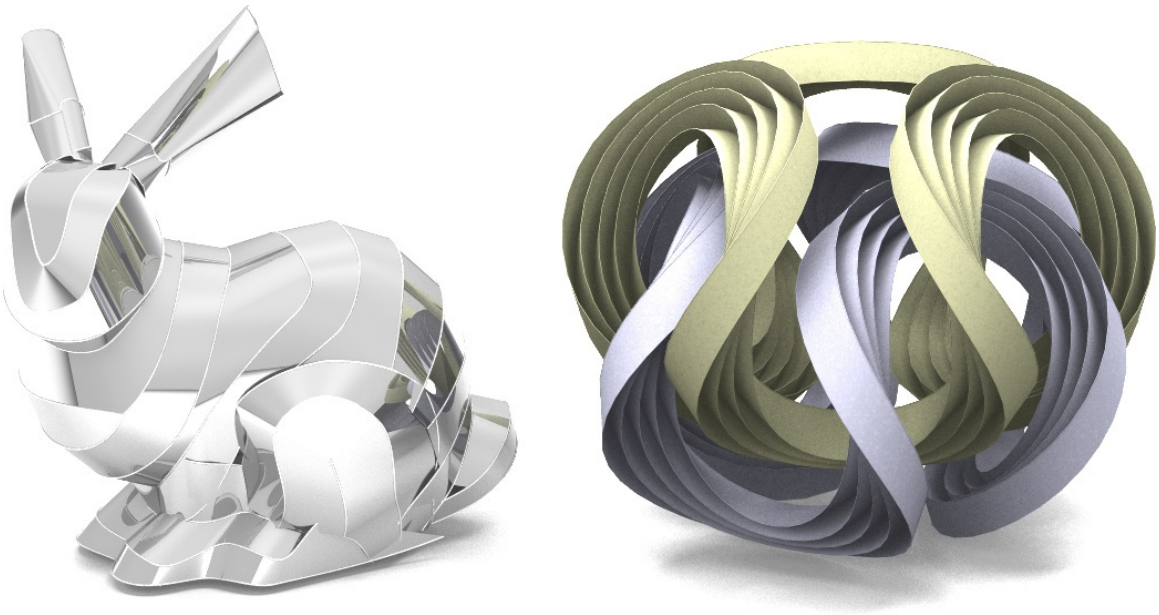


Figure 3.1: This chapter extends the approach of guided projection presented in the previous chapter to a general geometric modeling framework for developable surfaces. The discrete notion of face planarity for polyhedral meshes is extended to the developability which is property of contiuous surfaces. Such a generalized approach provides new methods for modeling and approximation with developable surfaces, as well as creating curved origami designs. The bunny at left shows approximation result with composite piecewise-developables of references surfaces. The curved origami on the right shows an interactive design inspired by the paper sculptures obtained by folding a sheet of paper along concentric rings, which go back to 1927 Bauhaus classes.

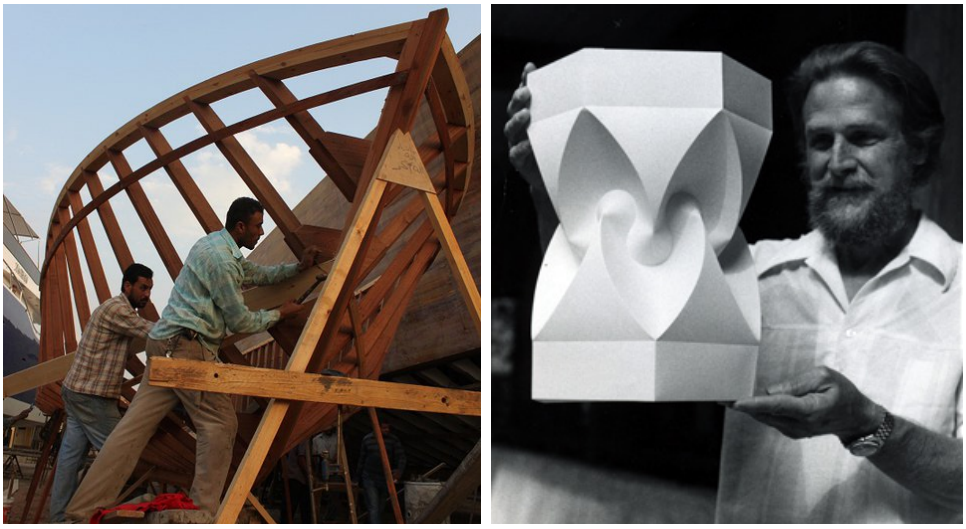


Figure 3.2: Left: boat builders in Alexandria are building the doubly curved hull by bending wooden strips, which is an ancient method that is still widely used to shape complex geometry. Right: David Huffman, a famous electrical engineer who had investigated the curvature behavior of curved folding [Huf76], and his Lantern featuring curved folding creases.

In practice, developable surfaces have a wide range of applications in both industry and architecture, as well as purely artistic designs. For industrial fabrication and architectural construction, developable surfaces are much simpler to manufacture compared with doubly curved surfaces. They could be processed by cutting, bending and merging sheet materials such as glass, metal, paper and wood. Besides, when developable panels, referred to as single curved panels by architects, are constructed by molding, the molds can be more efficiently reused compared with doubly curved panels.

The effectiveness of the application of developable surfaces has been demonstrated and proved throughout history. An example is lofting the hull of ships and boats. Ir is an old topic that has been studied empirically over several millennia through different civilizations. Despite originating from different geographic regions, they have invented similar approaches to construct doubly curved boat hulls with wooden strips. As

illustrated in Figure 3.2, boat builders in Alexandria, Egypt are paneling the doubly curved hull of a boat using wooden developable strips by traditional techniques.

The boundary between the practical applications and the artistic creations has never been restrictively defined. From an artistic point of view, origami is a field with a long and rich history that could be dated back to three centuries ago in Japan. It yet still being vividly developed with novel techniques that have been advanced greatly in the past three decades, especially through the recent invention of curved folding and novel tessellations. Most of the classical origami designs are created with straight line folds. David Huffman, an electrical engineer who is famous for his invention of the Huffman code, is arguably the first person who had studied origami featuring curved folds. A photo of David Huffman and his remarkable invention of the lantern with curved folds is shown in Figure 3.2, right. An architectural example with curved folds is shown in Figure 3.3. Folding a piece of paper into a nicely shaped object in space without tearing or stretching implies an isometric deformation between the plane and the surface. Such a surface is constituted with several smooth developable patches, merged along developable creases that satisfy special boundary conditions. Both the special boundary conditions for curved creases and the constraints to achieve isometric deformation are studied in this chapter.

3.2 Prior Work

There are many previous studies related to developable surfaces. In general, with different motivations, the geometry processing and the computer aided design communities have both studied the modeling of developable surfaces. While the former favors mesh based methods for flexibility and interactivity, the latter usually takes spline based approaches for higher precision. The advantages and shortcomings of

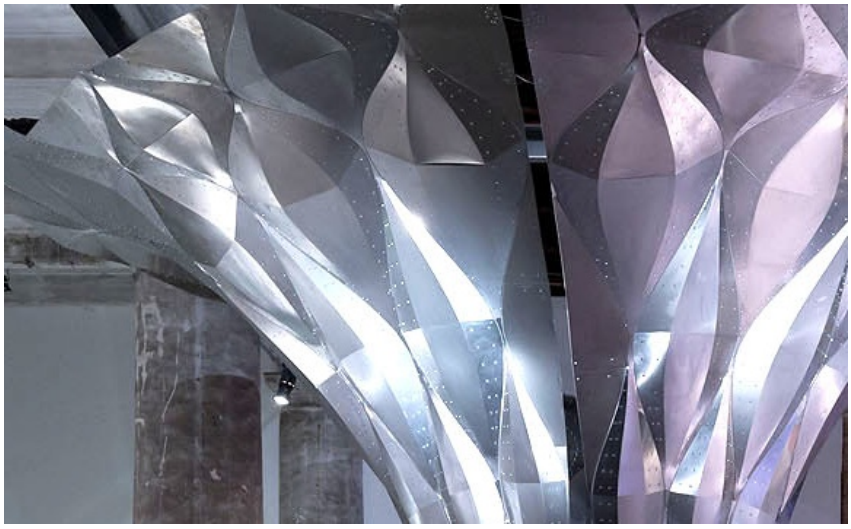


Figure 3.3: The "Arum" surface was designed by Zaha Hadid Architects in cooperation with Robofold for the 2012 Venice Biennale. The "peddles" are manufactured by folding metal sheets along curved creases by robots.

them have been very challenging to harmonize. In this section, the mesh based approaches are introduced first, followed by spline based methods. This short review also leads to the proposed methodology which combines the advantages of both these two families of methods, which is discussed in details in the next sections.

3.2.1 Mesh based approaches

Motivated by artistic creations such as curved origami designs (for a foundational theoretical study, see the paper by Huffman [Huf76]), there are many previous studies related to computational origami or paper sculptures. Most of them use meshes to represent discretized surfaces that are approximately developable, for example, [ML12], [MS04], [Tac10], [TM12] and [WC11]. The work of Kilian et al. [KFC+08] studies the digital reconstruction of curved origami based on laser scans. Mitani and Igarashi [MI11] propose an approach for creating curved origami designs by reflecting a single developable surface, with special types of curved folds. Demaine et

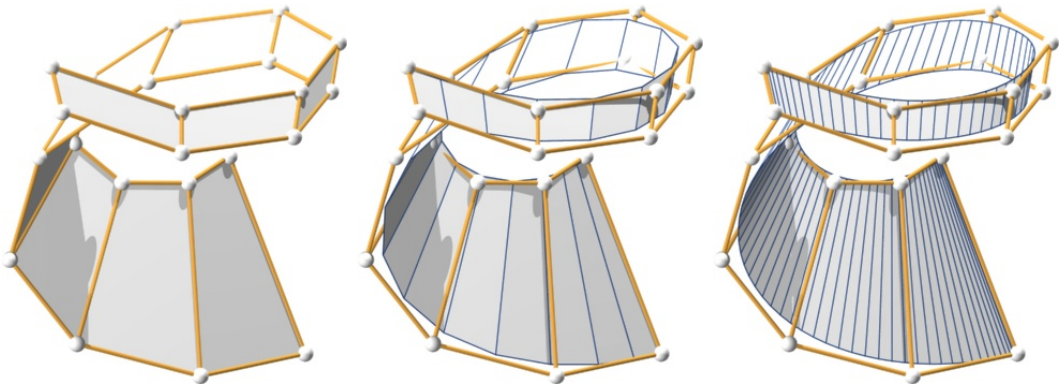


Figure 3.4: Liu et al. [LPW+06] created developable strips by consecutive subdividing and planarizing quadrilateral strips. Besides a highly nonlinear formulation for the planarity condition, such a purely mesh based formulation also cannot guarantee precise developability of a smooth surface.

al. [DDH+11] and Dias et al. [DDMS12] present several interesting examples which are also studied in this chapter. There are software packages developed to produce special kinds of curved origami designs. One such program is developed by Mitani [Mit12, MI11], which can be used to create rotational symmetric folding objects or curved folding objects by consecutive reflections.

As a natural extension of planar quadrilateral meshes, Liu et al. [LPW+06] propose a method to create developable strips by consecutively subdividing and planarizing quadrilateral strips, as shown in Figure 3.4. Besides a highly nonlinear formulation for the planarity condition, a purely mesh based formulation cannot guarantee precise developability of a smooth surface. Solomon et al. [SVWG12] present an approach to achieve the developability of surfaces and curved creases by maintaining the isometry between the surfaces in space and plane as shown in Figure 3.5. As this approach utilizes subdivision and optimization as well, it can only handle limited number of rulings, similar to other mesh based approaches. Besides sacrificing computational efficiency, the rulings and curved creases are fixed during manipulation, restricting

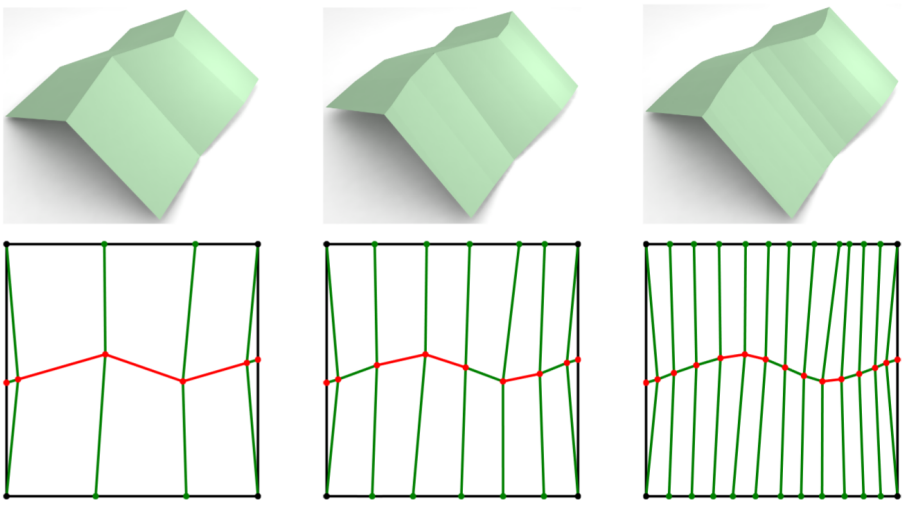


Figure 3.5: Solomon et al. [SVWG12] use developments to keep the developability conditions across creases. This approach utilizes subdivision in combination with optimization, similar to [LPW+06], with extra isometric constraints.

the variation of the ruling lines, therefore reducing the deformation flexibility of developable surfaces.

3.2.2 Spline based representations

The study of spline based developable surfaces has a long history, but it remains a fundamental challenge in the CAGD community. Compared with representations with discretized surfaces, spline based approaches can generate arbitrarily smooth surfaces, but the developability is difficult to achieve. These approaches can be further classified: representing the spline surface by primal variables (allowing users to work with the control vertices) directly, or creating developable surfaces through dual variables (allowing users to work with the control tangent planes).

Primal based approaches are intuitive to work with, but the constraints of the developability conditions are highly non-linear. This has resulted in them being re-

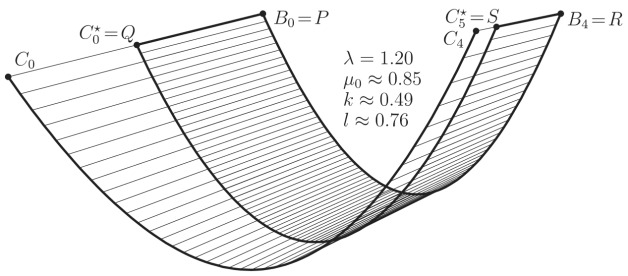


Figure 3.6: Aumann et al. [Aum91, Aum03] proposes constructive approaches to create simple developable surfaces like cones or cylinders based on Bézier (polynomial) patches with one given boundary curve.

garded as too challenging to solve in practice for general cases. Therefore, only special types of surfaces have been created through explicit constructions based on primal variables. Lang and Röschel [LR92] give a theoretical investigation of a system of cubic equations for the developability conditions of rational Bézier (polynomial) surfaces for all degrees. Maekawa and Chalfant [MC98] propose a method to create a very restricted type of spline developables: first, requiring two boundary curves of a developable patch to lie on parallel planes; further, the second planar curve has very limited degrees of freedom for further adjusting the shape of the surface. G. Aumann [Aum91], [Aum03] studies the special types of developable Bézier (polynomial) surfaces based on explicit constructions with one given boundary curve, as shown in Figure 3.6. In *Developable Bézier Function Surface*[2002], Chen and Wang attempt to create developable surfaces based on polynomial functions whose graphs are developable. However, surfaces of polynomial functions do not have singularities and therefore cannot represent general developable surfaces except cylinders. Hence, the conclusions of this paper is unfortunately challenged. Chu and Séquin [CS02] discuss the developability conditions of spline surfaces and derive explicit forms of the developability constraints for surfaces up to degree 3. Chu and Chen [CC04] continue this study and derive the number of degrees of freedom available for geometric modeling.

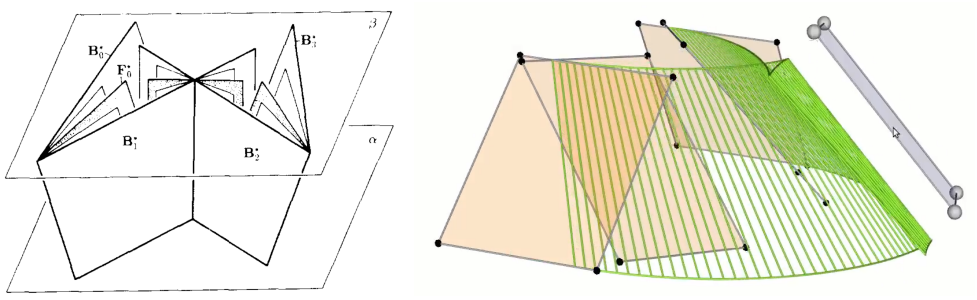


Figure 3.7: Left: Pottmann et al. [PF95] propose a method to generate developable surfaces by the movement of tangent planes, which can be regarded as points in the dual space. Right: Such dual based approaches enjoy developability automatically, but they are unintuitive to work with, and prone to generate singularities.

Based on the ideas of [LPW+06], Pottmann et al. [PSB+08] use splines for modeling developable surfaces to achieve approximated developability based on optimization, using integrals for the target functionals.

In contrast to primal based approaches, dual based methods create surfaces by the movement of planes in space, see, e.g., [PF95]. While the developability is achieved automatically with such approaches, the undefinedness of the boundary curves, the unintuitive manipulation interface, and the uncontrollability of singularities prevent their applicability for most design scenarios, see Figure 3.7 for example.

As demonstrated below, the proposed approach takes the advantages of both the preciseness of the spline based approach as well as the flexibility of the mesh based approach. The spline based representation is utilized for surface evaluation, and the constraints are kept through a mesh like discrete structure in the meanwhile.

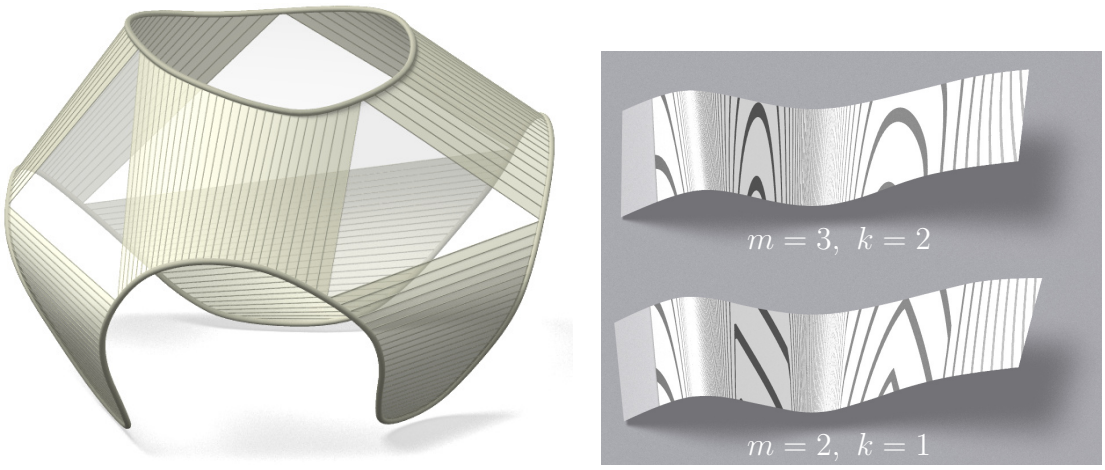


Figure 3.8: Developables decompose into planar pieces and ruled patches. Curvature continuity is relevant for surface quality since it becomes visible in reflection patterns. The right hand images show spline developables of varying degree m and smoothness C^k.

3.3 Developability Conditions

3.3.1 Simple developable patches

Each developable surface can be decomposed to simple developable ruled surfaces and planar patches, as demonstrated in Figure 3.8, left. In the following discussions, simple developable patches are introduced first, followed by multi-patches with or without special boundary conditions.

In order to create large patches of composite developable surfaces, a foundational study should be performed on the modeling of a single developable patch. Every simple developable patchs is a ruled surface patch. A ruled surface contains a family of lines and it can be parametrized by connecting two boundary curves $\mathbf{a}(u)$ and $\mathbf{b}(u)$:

$$\mathbf{s}(u, v) = (1 - v)\mathbf{a}(u) + v\mathbf{b}(u). \tag{3.1}$$

See Figure 3.9 for example.

Figure 3.9: A ruled surface (right) could be represented and parametrized by connecting two curves (left). It is developable if and only if each ruling defines a common tangent plane shared by all its points.

A ruled surface is developable if and only if each ruling defines a common tangent plane shared by its points. Alternatively, for any given u, the unit surface normal $\mathbf{n}(u, v)$ vectors are the same for all v. As the tangent planes on each ruling interpolate the tangent planes of the curve points, the developability conditions could be guaranteed by requiring $\mathbf{s}(u, 0) = \mathbf{a}(u)$ and $\mathbf{s}(u, 1) = \mathbf{b}(u)$ to share the same tangent plane or the same normal. Such conditions which can be expressed via different formulas as discussed below.

First, linear dependence of tangent vectors $\mathbf{a}'$, $\mathbf{b}'$ and the ruling $\mathbf{a} - \mathbf{b}$ can be expressed by the condition of their determinant in cubic Equation (3.2). Equivalently, by introducing an auxiliary normal vector field $\mathbf{n}(u)$, the condition could be expressed by three quadratic equations in Equation (3.3):

$$\mathbf{s} \text{ developable} \iff d(u) := \det(\mathbf{a} - \mathbf{b}, \mathbf{a}', \mathbf{b}') = 0 \tag{3.2}$$

$$\iff \langle \mathbf{n}, \mathbf{b} - \mathbf{a} \rangle = \langle \mathbf{n}, \mathbf{a}' \rangle = \langle \mathbf{n}, \mathbf{b}' \rangle = 0, \tag{3.3}$$

for all parameter values u.

For simplicity and efficiency, it is useful to introduce two families of auxiliary

points $\mathbf{a}_0^{(1)}(u)$, $\mathbf{a}_1^{(1)}(u)$ which span the curve's tangent line at the points $\mathbf{a}(u)$, and similarly for the curve $\mathbf{b}(u)$. A straightforward choice of the auxiliary points would be $\mathbf{a}_0^{(1)} = \mathbf{a}$ and $\mathbf{a}_1^{(1)} = \mathbf{a} + \mathbf{a}'$. The more interesting alternative is demonstrated in Figure 3.10. In both the cases, the developability condition is expressed as

$$\begin{aligned} &\mathbf{s} \text{ developable} \iff \mathbf{a}_0^{(1)}, \mathbf{a}_1^{(1)}, \mathbf{b}_0^{(1)}, \mathbf{b}_1^{(1)} \text{ co-planar} \iff \\ &\langle \mathbf{n}, \mathbf{b}_1^{(1)} - \mathbf{a}_1^{(1)} \rangle = \langle \mathbf{n}, \mathbf{a}_1^{(1)} - \mathbf{a}_0^{(1)} \rangle = \langle \mathbf{n}, \mathbf{b}_1^{(1)} - \mathbf{b}_0^{(1)} \rangle = 0. \end{aligned} \tag{3.4}$$

These conditions are required for all the parameters u, which seemingly require the problem to be infinitely dimensional. Fortunately, as demonstrated in §3.3.1, a finite number of constraints required at sampled parameters are sufficient to guarantee that the constraints are satisfied everywhere. The conditions (3.3) and (3.4) are used in our computations because they lead to quadratic equations expressing developability. At the same time, the original cubic conditions provided by (3.2) is useful for counting degrees of freedom.

Degenerate cases

There are several special cases worth mentioning. When one of the two boundary curves degenerates to a single point, $\mathbf{b}(u) = \text{const.}$, the surface $\mathbf{s}(u, v)$ is a cone with base curve $\mathbf{a}(u)$ and apex coinciding with $\mathbf{b}_0^{(1)}(u) = \mathbf{b}_1^{(1)}(u) = \mathbf{b}(u) = \text{const}$. Similarly, when $\mathbf{a}(u) \equiv \mathbf{b}(u)$, the surface $\mathbf{s}(u, v)$ is a cylinder. There is an interesting fact. The cylinders have the same number of degrees of freedom as, do the spline curves, which is more than general developable patches. This is due to the fact that the cylinders correspond to the extraneous sheets of the constraint manifold, similar to polyhedral meshes degenerated to a plane, which have more degrees of freedom than polyhedral meshes in space. Similar arguments can be made for another degenerate case for the

planar ruled surfaces created by connecting two curves in a common plane, which are also seemingly more flexible.

Spline developables

In order to take advantage of the finite dimensionality of curve representation and constraint formulation (discussed below), polynomial and piecewise-polynomial curves are used. These can be conveniently expressed by Bézier curves or B-spline curves. The parameter interval $[u_0, u_n] = [0, 1]$ is used for Bézier curves. The interval is further subdivided into n nonempty sub-intervals $[u_0, u_1] \cup [u_1, u_2] \cup \ldots \cup [u_{n-1}, u_n]$ for B-spline curves. *Spline curves* are curves whose coordinate functions are *spline functions*. For B-spline curves, B-spline functions which exhibit C^k smoothness and

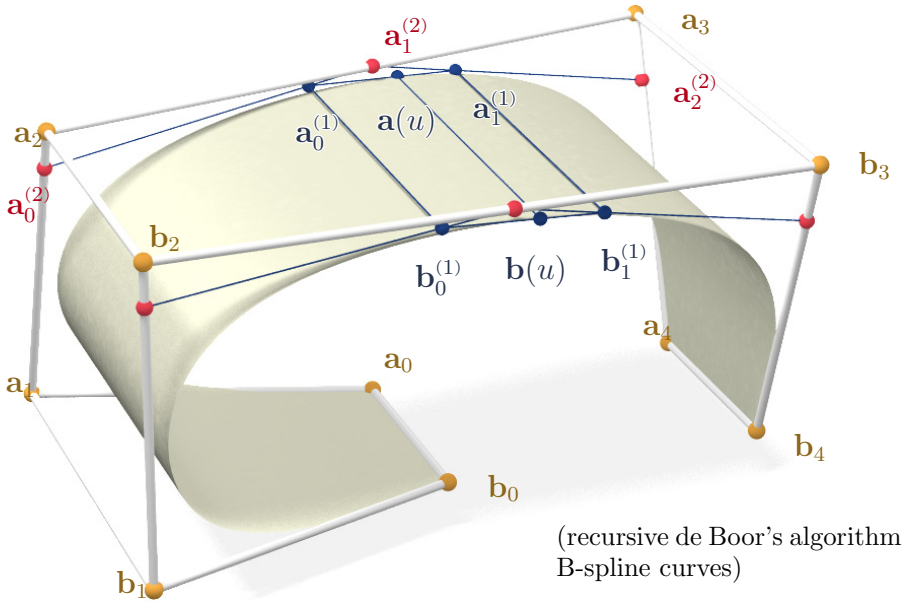


Figure 3.10: A spline curve $\mathbf{a}(u)$ is defined by its control points $\mathbf{a}_0$, $\mathbf{a}_1, \ldots$. It is equipped with auxiliary first derivative points $\mathbf{a}_0^{(1)}(u)$, $\mathbf{a}_1^{(1)}(u)$ which span the tangent line, and second derivative points $\mathbf{a}_0^{(2)}(u)$, $\mathbf{a}_1^{(2)}(u)$, $\mathbf{a}_2^{(2)}(u)$ which span the osculating plane of the curve. These auxiliary points are computed with de Boor's algorithm. Developability of the ruled surface defined by curves $\mathbf{a}$, $\mathbf{b}$ is equivalent to coplanarity of $\mathbf{a}_0^{(1)}$, $\mathbf{a}_1^{(1)}$, $\mathbf{b}_0^{(1)}$, $\mathbf{b}_1^{(1)}$.

which are polynomial of degree m in each subinterval ($k < m$) are considered. To describe a *closed* B-spline curve, all derivatives up to order k are required to be equal at the two interval ends for $u = u_0$ and $u = u_n$.

B-spline curves are linear combinations of the B-spline basis functions $N_i(u)$:

$$\mathbf{a}(u) = \sum\nolimits_i \mathbf{a}_i N_i(u) \tag{3.5}$$

The points $\mathbf{a}_0, \mathbf{a}_1, \ldots$ are the control points of the curve $\mathbf{a}(u)$. The basis functions $N_1(u), N_2(u), \ldots$ spanning a particular spline space can be constructed through well known procedures, see, e.g., [dB78].

For any u, evaluation of the point $\mathbf{a}(u)$ could alternatively be performed through a more geometric procedure with de Boor's algorithm as demonstrated in Figure 3.10). When $n = 1$, B-spline curves reduce to Bézier curves with single parameter intervals, and de Boor's algorithm reduces to de Casteljau's algorithm.

De Boor's algorithm iteratively computes linear combinations of points. It starts with the control points in the first round of iteration, and ends up with the curve points $\mathbf{a}(u)$ in the last round. In the next-to-last round, it gives auxiliary "first derivative" points $\mathbf{a}_0^{(1)}(u)$, $\mathbf{a}_1^{(1)}(u)$, which can be used for the developability condition (3.4) as they span the tangent line of the curve in the point $\mathbf{a}(u)$. Before the next-to-last round, it gives the second derivative points $\mathbf{a}_0^{(2)}(u)$, $\mathbf{a}_1^{(2)}(u)$, $\mathbf{a}_2^{(2)}(u)$, which span the osculating plane of the curve and are useful for expressing special boundary conditions merging patches which as discussed below.

Computation with simple spline developables

For each simple B-spline ruled surface patch, the developability is characterized by vanishing of the degree $3m - 3$ polynomial $d(u)$ in (3.2) for each parameter interval.

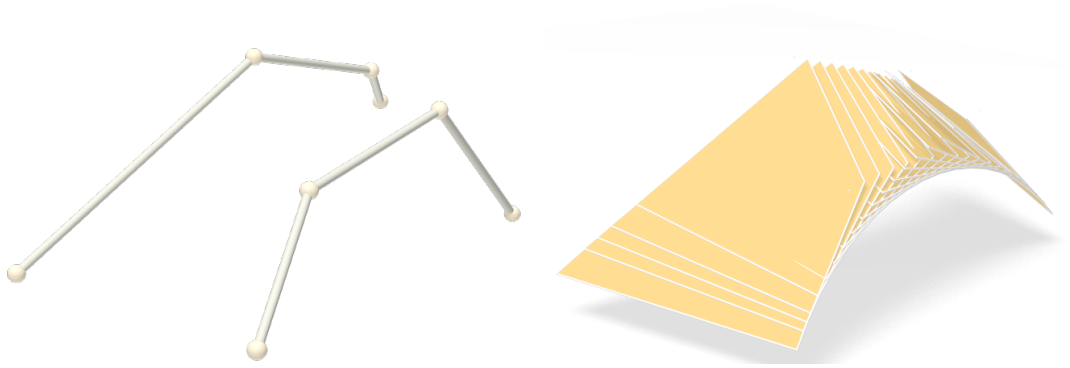


Figure 3.11: The control vertices of the two spline curves (left) and the auxiliary points of the tangent quadrilaterals (right) are kept as variables throughout the computation while the surface points of the ruled surface (see Figure 3.9, right, for example) are not necessary. The constraints imposed for achieving the developability condition of this simple patch include the planarity of the auxiliary quadrilaterals, and the predefined linear relations between the auxiliary vertices and the control vertices given by the de Boor's algorithm.

It is sufficient to enforce the developability condition (3.4) for $3m-2$ different values of u within each parameter interval. In computation, B-spline control points of the curves $\mathbf{a}$ and $\mathbf{b}$, as well as the auxiliary points $\mathbf{a}_0^{(1)}(u)$, $\mathbf{a}_1^{(1)}(u)$, $\mathbf{b}_0^{(1)}(u)$, $\mathbf{b}_1^{(1)}(u)$ are kept as variables through predefined linear relations given by the de Boor's algorithm, as shown in Figure 3.11. The normal vectors $\mathbf{n}(u)$ could be introduced as independent unit vectors, or samples from a dual B-spline with additional normal control points related by linear relations as well.

3.3.2 Multi-patch developable surfaces

Based upon the simple developables, general developable surfaces could be constructed by merging simple ruled developable patches and planar polygons. The complete system of the computation therefore stores all the variables representing individual patches and solves for them simultaneously. Figure 3.13 visualizes the auxiliary quads and control vertices used in a simple multi-patch developable surface.

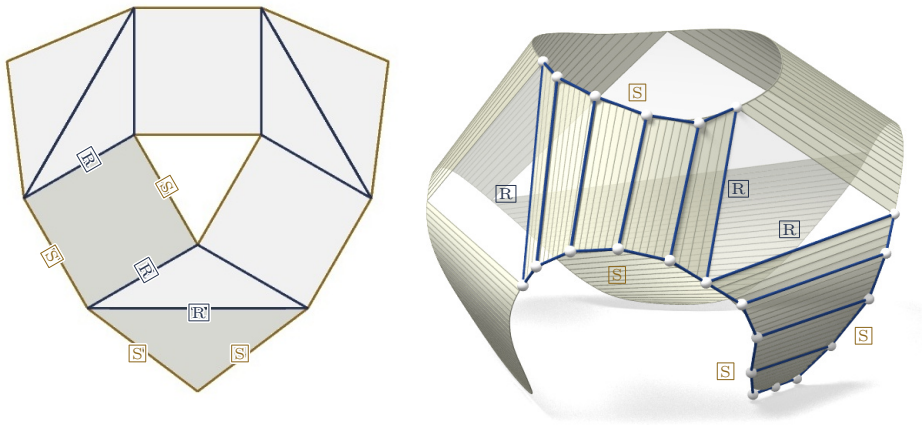


Figure 3.12: The combinatorial setup of a composite seamless developable is stored in a mesh, where edges are labelled either R for "ruling" or S for "spline". Faces accordingly correspond to spline developables, cones, or planar parts. The example shown here has no cones. There is smooth transition between surfaces along all nine R-edges, and between boundary curves in all 12 vertices.

The efficiency of the algorithm allows interactive editing with composite developable surfaces.

To represent the combinatorial relations of simple patches and polygons constituting a general composite developable surface, a coarse mesh is introduced, as shown in Figure 3.12. Every face represents a patch and each edge corresponds to either a ruling line Ror a spline S. There are four possible types of faces for such a coarse mesh:

- SRSR: a typical spline developable formed by connecting two splines with two rulings on the ends;
- SRS: a similar spline developable with a ruling degenerated into a point;
- SRR: a cone with a spline boundary degenerated to a point which does not require developability conditions to be imposed;
- RR...R: a planar polygonal face.

The SRS and the SRR patches can be represented by SRSR with coinciding ruling ends or control points. In practice, they are regarded as separate types of

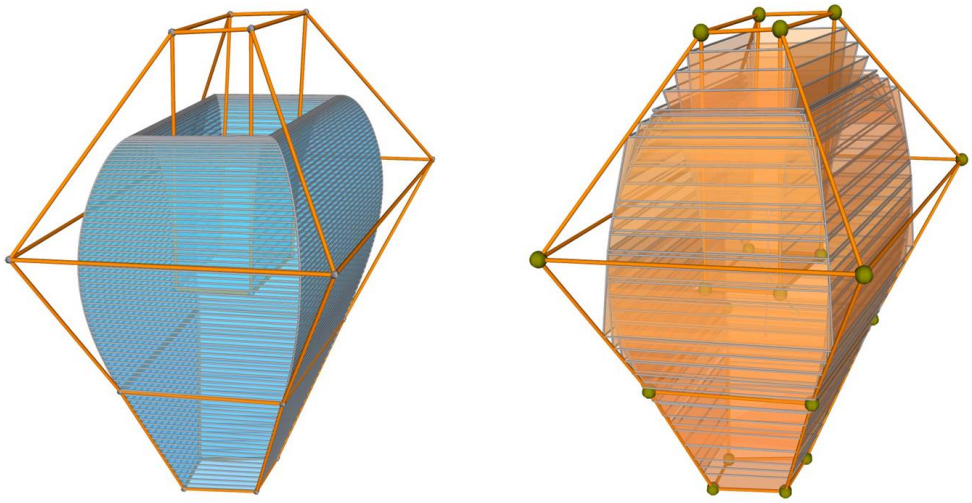


Figure 3.13: The developability condition of a multi-patch composite developable surface (left) can be imposed by the auxiliary quads introduced for each spline patch (right). Similar to the simple patches shown in Figure 3.11, only the control vertices and the auxiliary vertices of the tangent quads are necessary to be kept as variables througout the computation. The discretized surfaces, which can be evaluated directly based on the control structures, are only required for visualization, or additional conditions imposed and discussed below.

surfaces for computational efficiency as the redundant variables are omitted.

Most of the practically useful composite developable surfaces can be represented by finitely many simple patches of the aforementioned four types. The developability conditions could be naturally derived based on simple patches through the planarity of the auxiliary tangent quadrilaterals, as illustrated in Figure 3.13. Over-fragmented developable surfaces may constitute an infinite number of simple patches. However, these surfaces, with few fabrication benefits, are generally discarded in the context of computational design. Several industrial examples designed with the proposed approach are illustrated in Figure 3.14 and Figure 3.15.

With a given coarse mesh, the control vertices, auxiliary vertices, as well as the surface points can be subsequently constructed or evaluated and applied in compu-

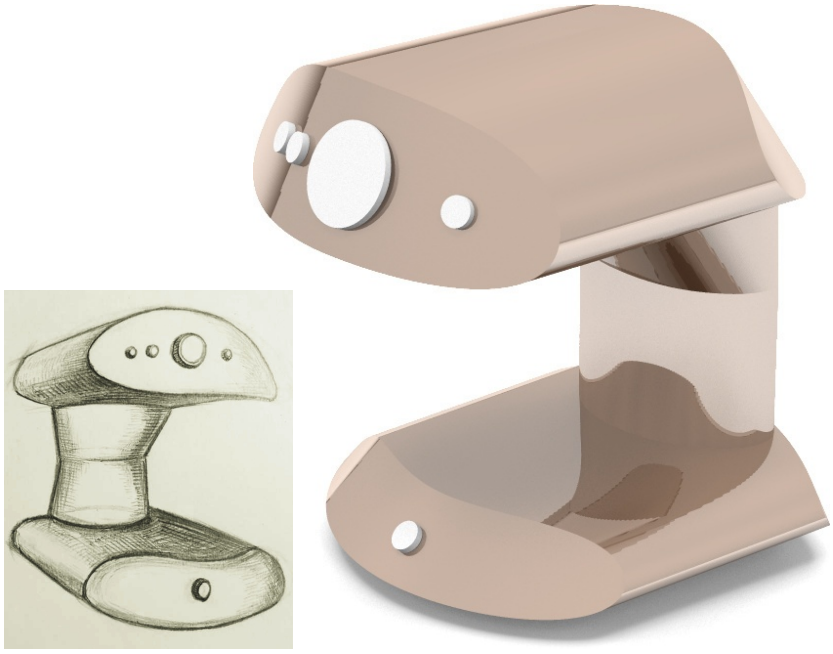


Figure 3.14: A piecewise-developable model created with the proposed approach based on an artist's sketch.

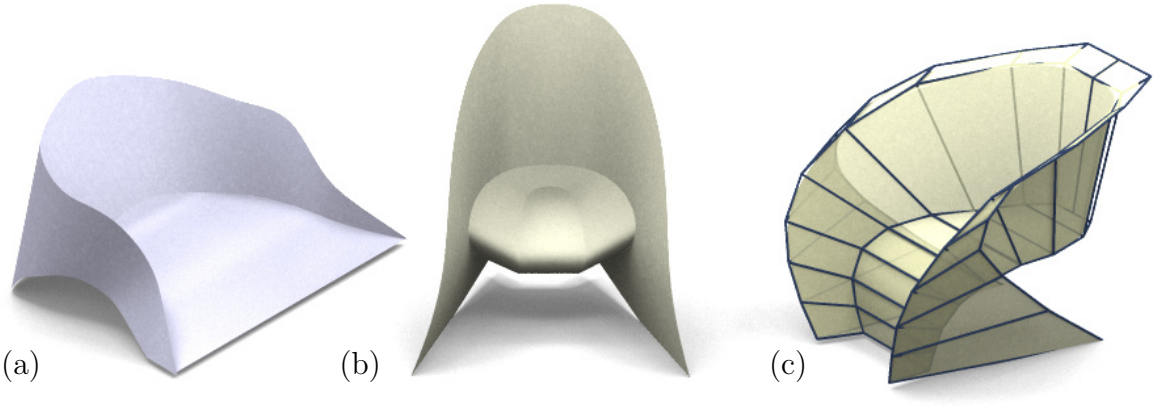


Figure 3.15: These surfaces are the product of interactive modeling. They all have the same combinatorics (in the sense of Figure 3.12). In subfigure (c), control points are connected by blue edges.

tation. For mainly visualization purposes, surface meshes could be evaluated and interactively updated, based on the computed control structures. However, the computational efficiency for developability is independent of the number of surface points of rulings sampled. This is different from previous approaches based on discrete representations.

3.3.3 Developable approximation

To allow the fabrication of doubly curved surfaces by developable patches, approximating the freeform shapes by developable surfaces, as shown in Figure 3.16, is often a desired function.

Approximation with developable surfaces can be achieved by minimizing the distances between the targets and the spline surfaces. Figure 3.17 demonstrates a typical application where a bunny model is approximated by a piece-wise developable surface. It is observed that minimizing the differences of normal vectors together with the distances improves the convergence of approximation, thereby validating the efficiency of our strategy to introduce auxiliary variables, from a different perspective.

All the approximation targets are regarded as point clouds, preprocessed through a thinning process that is is very similar to the Poisson sampling. During each round, a sampled point $\mathbf{q}_j$ is chosen, all the points within the ball a fixed radius r centered at $\mathbf{q}_j$ are used to estimate a normal vector $\mathbf{m}_j$ and subsequently discarded. The measure

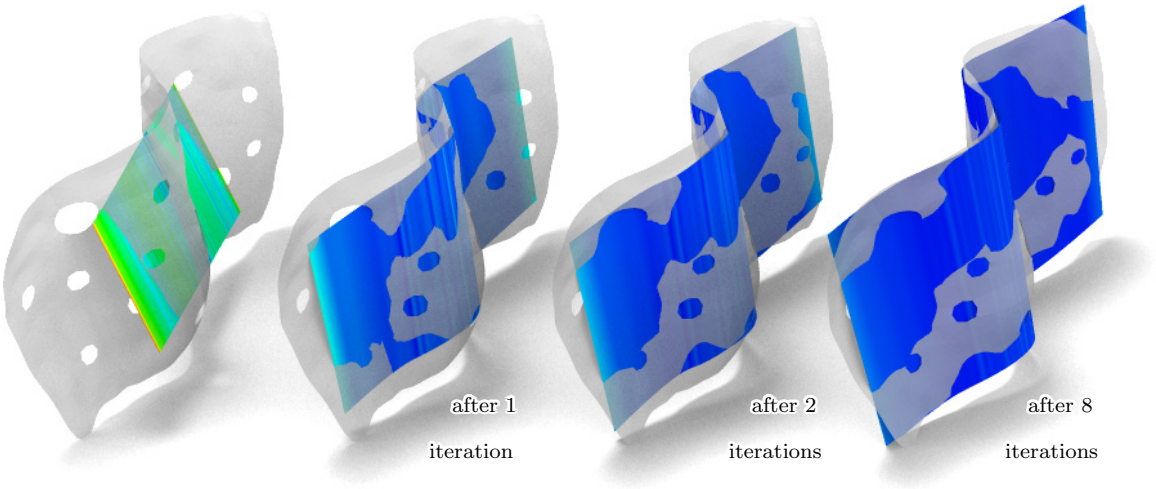


Figure 3.16: A simple approximation problem, where a reference shape (point cloud) is approximated by a developable strip. The equations we use to express proximity make the strip grow so as to cover as much of the reference shape as possible. The number of iterations refers to guide projection.

of proximity constituts of three parts.

First, the most straightforward measure is the distance between a sampled point $\mathbf{q}_j$ and the corresponding footpoint $sw(u_j, v_j)$ expressed as a linear combination of the control points:

$$\mathbf{s}(u_j, v_j) - \mathbf{q}_j = \mathbf{o}. \tag{3.6}$$

The second measure is the orthogonality between the reference normal $\mathbf{m}_j$ with respect to the vectors spanning the tangent plane at the foot point $sw(u_j, v_j)$:

$$\begin{aligned} \langle \mathbf{m}_j, \mathbf{a}(u_j) - \mathbf{b}(u_j) \rangle &= 0 \\ \langle \mathbf{m}_j, \mathbf{a}'(u_j) \rangle &= 0. \\ \langle \mathbf{m}_j, \mathbf{b}'(u_j) \rangle &= 0. \end{aligned} \tag{3.7}$$

The last term is to to further discourage the distance along the direction that is orthogonal to the surface:

$$\langle \mathbf{s}(u_j, v_j) - \mathbf{q}_j, \mathbf{n}(u_j) \rangle = 0. \tag{3.8}$$

In practice, as this distance measure prevents tangential movement, it is only given a very small weight, while the latter two terms are set to comparatively higher parameters.

Merging patches

For complex surfaces, it is essential to provide users the ability to approximate surfaces with desired connectivity of surfaces. To achieve this, an interactive framework proposed allows users to paint individual strokes on the doubly curved surface for simple patches. Then, with user specification, the individual patches could be merged

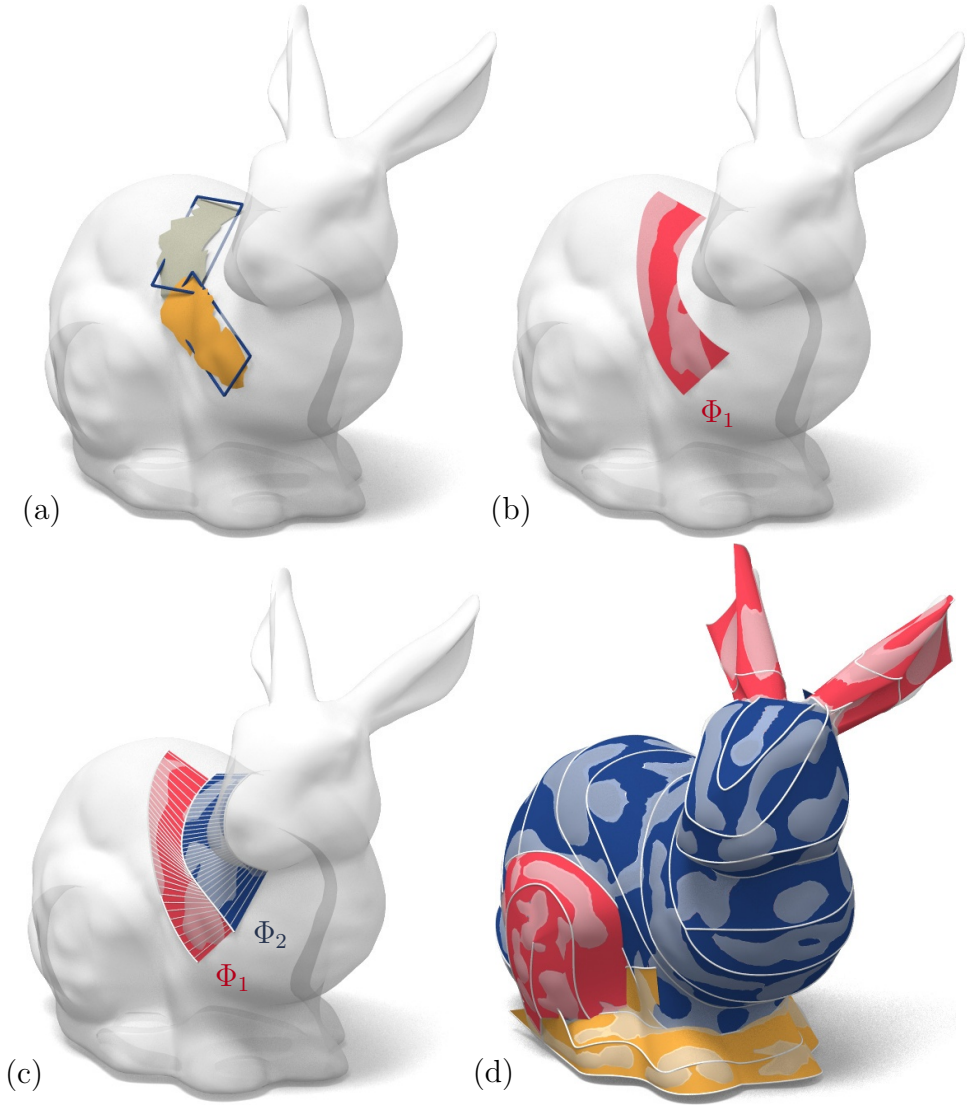


Figure 3.17: Approximating the Stanford bunny by developable strips. (a) The user marks two areas on the bunny intended to be approximated by cubic developables. PCA is used to fit initial flat strips, which are visualized as rectangles. (b) The two strips obtained in the previous step are merged by putting together their respective control point sequences (averaging the end control points) and approximating again. (c) Having produced two strips side by side, they are subjected to the approximation procedure, with proximity of boundaries as an additional optimization goal. Such strips serve to initialize a composite surface. (d) The bunny is approximated by 6 composite surfaces shown in different colours. Figure 3.1 displays the same surface.

together into a water time piece as a composite piece-wise developable surface, followed by an approximation procedure globally. Figure 3.17 shows such an example.

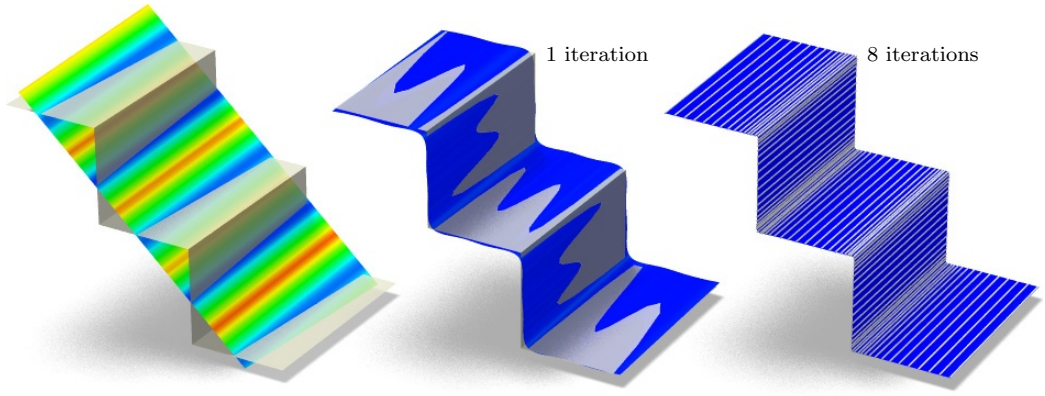


Figure 3.18: Spline developables can approximate shapes with edges, if the normal spline is not enforced to have higher order smoothness and if $\mathbf{n}(u_j)$ in (3.8) is considered fixed in each iteration of constraint solving. This image colour codes the distance of a developable spline strip from the reference shape.

Approximation with Sharp features

A widely held belief towards spline based representation is that the general smooth splines have limited capability to represent or resolve sharp features. However, there are two ways to overcome such restrictions. One possible approach is to introduce multiplicity for the knot vectors of the spline representation. Another, more natural solution, is to allow control points to coincide. Figure 3.18 demonstrates such an example based on the automatically collocated control points.

3.4 Curved Origami

When developable patches are merged together, special types of boundary conditions can be further required. A typical and practical requirement is the global developability, where multiple patches in space could be flattened to be planarized together without tearing or stretching. Such a requirements is necessary for designing curved origami. Figure 3.19 illustrates such shapes which are made by folding single sheets

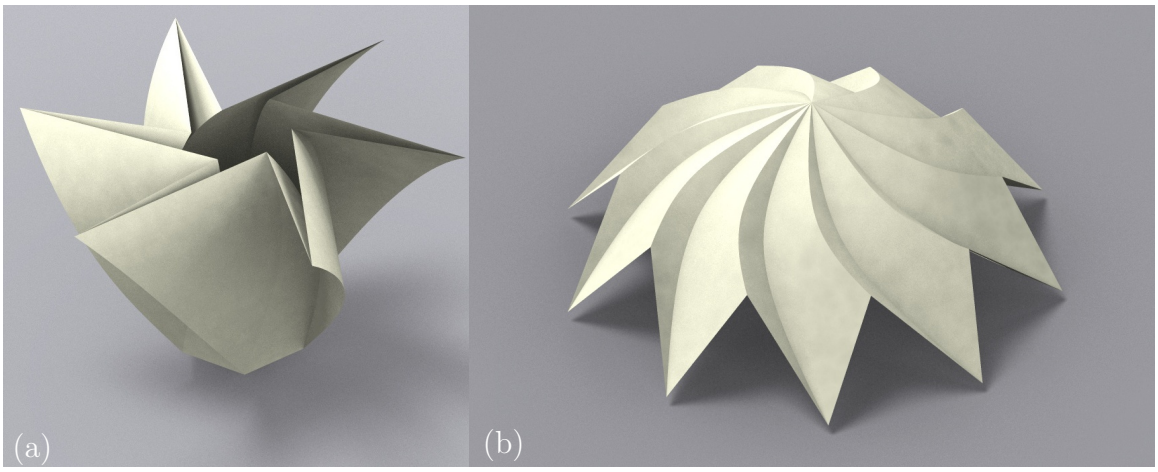


Figure 3.19: Shapes foldable from a single sheet of paper consist of ruled patches. Each patch is developable. The smooth creases enjoy the "curved fold" property. The angle sum around each vertex equals 2π. These three conditions are sufficient for local developability.

of paper with curved creases and developable vertices.

3.4.1 Curved folding boundary conditions

In curved origami, a curved spline crease $\mathbf{c}(u)$ which merges two strips Φ^{Left} and Φ^{Right} could be developed to a single planar curve. Therefore, the geodesic curvatures $\kappa_g^{\text{Left}}(u)$ and $\kappa_g^{\text{Right}}(u)$ for Φ^{Left} and Φ^{Right} respectively are the same:

$$\kappa_g^{\text{Left}}(u) = \kappa_g^{\text{Right}}(u). \tag{3.9}$$

At corresponding points, the curvatures of a developed planar curve equal the geodesic curvatures κ_g of the spatial curve, which are related to the spatial curve's curvatures κ by $\kappa_g = \kappa \sin\theta$. Here, the angles, θ, are the angles between the surface tangent planes and the curve's osculating planes. As the geodesic curvatures are the same for the two patches, the spline crease $\mathbf{c}$'s osculating planes bisect the tangent planes of Φ^{Left} and Φ^{Right}, with unit normal vectors $\mathbf{n}^{\text{Left}}(u)$ and $\mathbf{n}^{\text{Right}}(u)$ respectively.

As demonstrated in Figure 3.10, the de Boor's algorithm provides auxiliary second

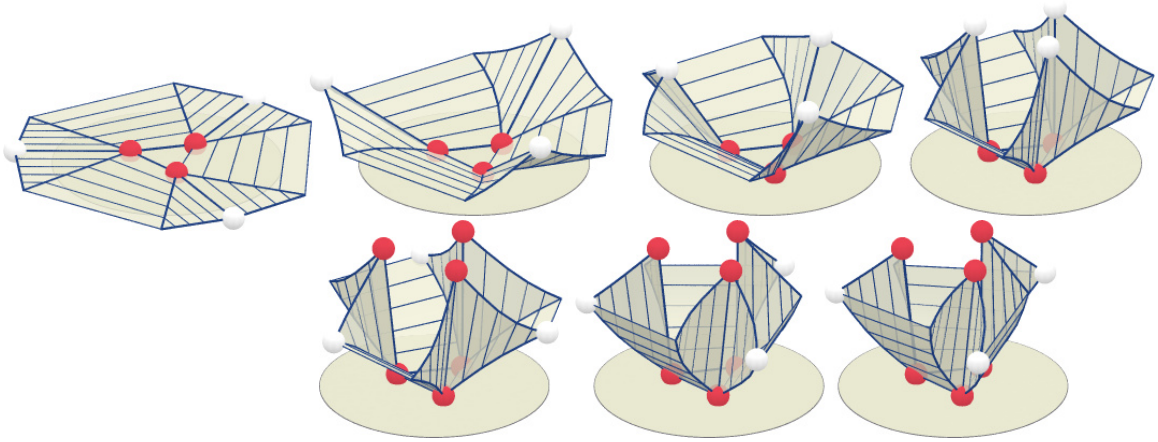


Figure 3.20: Interactive modeling under curved-fold side conditions. Starting with a flat composite developable, on top we show dragging selected (white) vertices upwards while other vertices (red) remain fixed. The bottom row illustrates rotating selected vertices around a vertical axis. Control points are connected by blue edges.

derivative points of the spline $\mathbf{c}$ which span the osculating planes of $\mathbf{c}(u)$. By denoting these second derivative points as $\mathbf{c}_0^{(2)}(u)$, $\mathbf{c}_1^{(2)}(u)$, $\mathbf{c}_2^{(2)}(u)$, the curved folding conditions could be written as:

$$\begin{aligned} \langle \mathbf{c}_0^{(2)} - \mathbf{c}_1^{(2)}, \mathbf{n}^{\text{Left}} + \mathbf{n}^{\text{Right}} \rangle = 0, \\ \langle \mathbf{c}_1^{(2)} - \mathbf{c}_2^{(2)}, \mathbf{n}^{\text{Left}} + \mathbf{n}^{\text{Right}} \rangle = 0. \end{aligned} \tag{3.10}$$

These conditions are incorporated in the computation together with the linear relations between the second derivative points and the control points. Figure 3.20 illustrates interactive design with curved folding boundary conditions .

Geodesic boundary conditions

Closely related to curved folding conditions, the geodesic boundary conditions also require the osculating planes of the boundary curves to bisect the tangent planes of the adjacent spline patches, in the directions that are orthogonal to the bisecting

planes appeared in the curved folding conditions.

$$\begin{aligned} \langle \mathbf{c}_0^{(2)} - \mathbf{c}_1^{(2)}, \mathbf{n}^{\text{Left}} - \mathbf{n}^{\text{Right}} \rangle &= 0, \\ \langle \mathbf{c}_1^{(2)} - \mathbf{c}_2^{(2)}, \mathbf{n}^{\text{Left}} - \mathbf{n}^{\text{Right}} \rangle &= 0. \end{aligned} \tag{3.11}$$

In this case, the planar boundaries of the developed patches are reflectively symmetric to each other, which provide further benefits for the panelization of architectural freeform skins (see, e.g., [PSB+08]). Figure 3.21 presents an example.

Boundaries with straight development

A boundary $\mathbf{c}$ on a surface patch Φ has a straight development if it has a zero geodesic curvature. Similar to Equaitions (3.10) and (3.11), such conditions could be imposed by further introducing auxiliary normals, $\mathbf{n}^{\text{osc}}$, for the osculating planes of the curve $\mathbf{a}(u)$, and then requiring their orthogonality to the surface normals $\mathbf{n}$.

$$\begin{aligned} \langle \mathbf{c}_0^{(2)} - \mathbf{c}_1^{(2)}, \mathbf{n}^{\text{osc}} \rangle &= 0, \\ \langle \mathbf{c}_1^{(2)} - \mathbf{c}_2^{(2)}, \mathbf{n}^{\text{osc}} \rangle &= 0, \\ \langle \mathbf{n}^{\text{Left}}, \mathbf{n}^{\text{osc}} \rangle &= 0. \end{aligned} \tag{3.12}$$

To add either property to the computational setup, the conditions (3.10) and (3.11) are required for sufficiently many values of u, each time, adding the objects involved as auxiliary variables and adding as equations the linear defining relations of $\mathbf{c}_i^{(2)}$, an appropriately relabelled Equation (3.4) which defines the normal vectors $\mathbf{n}^{\text{Left}}$, $\mathbf{n}^{\text{Right}}$. Finally the normalization constraints $\|\mathbf{n}^{\text{Left}}\|^2 = \|\mathbf{n}^{\text{Right}}\|^2 = 1$ are required.

Remark: In § 3.3.1 it was argued that enforcing(3.2) for $3m-2$ values of u in each parameter sub-interval implies validity of (3.2) for all u. The corresponding number of evaluations for (3.10) and (3.11) would be too high, so these conditions are enforced

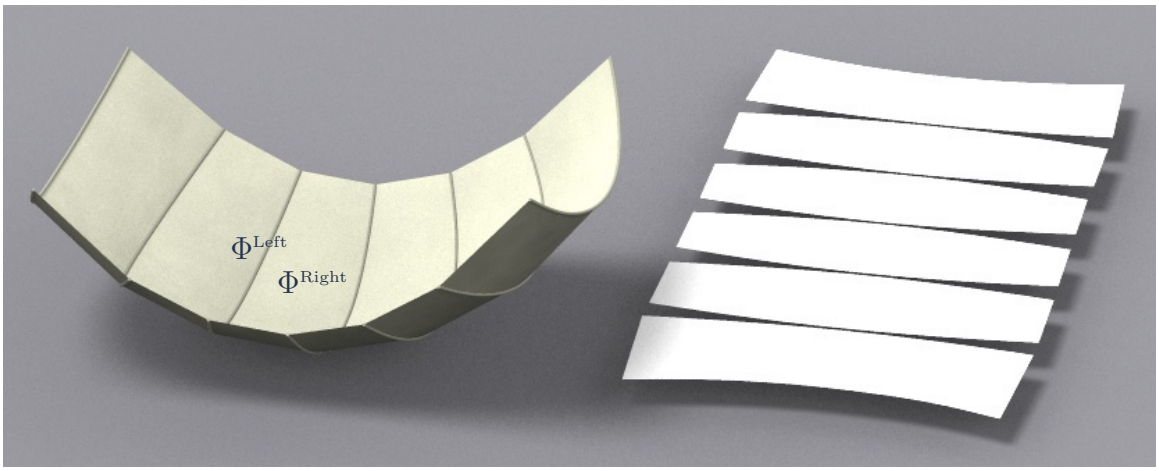


Figure 3.21: Here, unfolding neighbouring strips Φ^{Left}, Φ^{Right} maps their common boundary to planar curves which are mirror reflections of each other. This "geodesic" property makes the developments of strips straighter, which is useful for materials like wood [MK12].

only numerically.

3.4.2 Isometric manipulation

Closely related to curved origami design is the isometric manipulation of developable surfaces in space, i.e., with fixed or constrained development, as shown in Figure 3.22. For artistically oriented designers, such capabilities provide similar experiences to physical experiments of real sheet materials. For engineers, in parallel with designing spatial surfaces, isometric manipulation allows the control of the development constituted of planar patches before further fabrication, e.g., by cutting, bending and welding or cladding.

In order to achieve isometric conditions during interactive manipulation, a discretized development is maintained together with a corresponding discretized surface mesh in space. The discretized spatial mesh (V, E, F) is initialized based on de Boor's algorithm for each individual spline surface $\mathbf{s}(u, v)$ by predetermined samples of parameters u for the two boundary curves, $\mathbf{s}(u_i, 0) = \mathbf{a}(u_i)$ and $\mathbf{s}(u_i, 1) = \mathbf{b}(u_i)$. The

Figure 3.22: Isometric manipulation of a strip to create a Mobius band. The texture coordinates are interactively updated by the coordinates of the points on the development which is isometric to the spatial mesh and constrained on a given rectangle.

faces of the spatial mesh are mostly quads or triangles, and are almost planar for developable spline surfaces. During computation, the linear relations derived from (3.5) between the surface points and the control points are kept as additional constraints.

It is worth mentioning that the discretized surface meshes required here are not needed for guaranteeing the developability conditions discussed above. Compared with the design of developables without considering the developments, enforcing isometric conditions introduces considerably more variables and equations, but can still be accommodated at interactive rates for the examples demonstrated in this chapter. Due to the discretization errors, isometric manipulation cannot achieve the same precision compared to the developability conditions which can be guaranteed precisely. Nevertheless, the precision of isometric conditions can be improved through refinement.

There are two major tasks related to isometric deformation: the initialization of the development before interactive manipulation, as well as the maintenance of the isometric relation between the pair of spatial and planar meshes.

Development initialization

The development $(\bar{V}, \bar{E}, \bar{F})$ of the spatial mesh (V, E, F) needs to be initialized based on the spatial discretization. When the manipulation of a developable design starts from the plane, the initialization of the development is trivial. For more general cases, there are two common ways to create initial developments with given globally developable surfaces.

A first approach is to start with an arbitrary planar quad of the surface mesh, and subsequentially glue adjacent faces congruent to the spatial faces on the same plane. Such a construction is consistent regardless of the choice of the initial face or the sequence of construction sequence, which is guaranteed by the global developability. Instead of such an approach based on explicit constructions, the initial development could also be computed in a numerical way by flattening onto a plane via minimizing

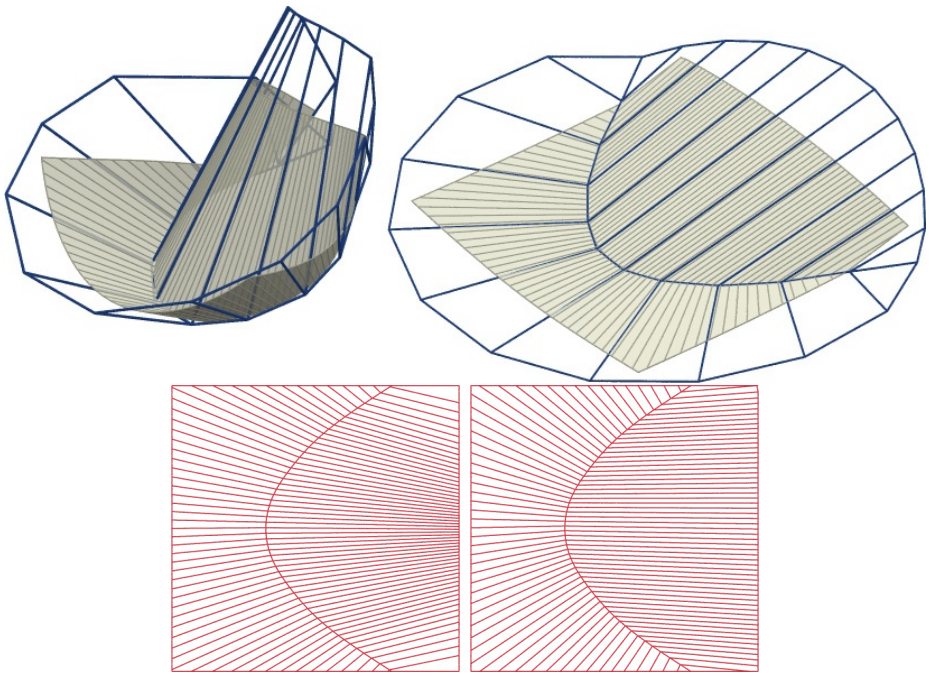


Figure 3.23: Isometric deformation of a rectangular paper folded along a given curved crease. To achieve better flexbility by accomodating greater ruling changes, the model is embeded to a disk like domain for computation.

the closeness of adjacent patch normals and maintaining the congruency of individual planar quads.

Isometric conditions

As a surface evolves, the isometric relation between the discretized surface in space (V, E, F) and the associated development $(\bar{V}, \bar{E}, \bar{F})$ has to be maintained. This could be naturally achieved by enforcing the isometric conditions of each pair of corresponding faces to the guided projection procedure, by equating the edges and diagonals:

$$\|\mathbf{v}_i - \mathbf{v}_j\|^2 = \|\bar{\mathbf{v}}_i - \bar{\mathbf{v}}_j\|^2 \quad (1 \leq i < j \leq n). \tag{3.13}$$

Here, $\mathbf{v}_j$ and $\bar{\mathbf{v}}_j$ are vertices of the spatial and developed meshes respectively, maintained as variables during computation. Figure 3.23 and Figure 3.25 illustrate two examples with isometric deformation.

Developability of vertices

Isometric conditions could already be applied to ensure the vertices are developable approximately. However, there is a more precise alternative to achieving this which is not subject to discretization errors, as illustrated in Figure 3.24.

Overfolding avoidance

Isometric constraints alone cannot guarantee the bijectivity between the spatial and the developed surfaces due to the possibility of over-foldings. There are different choices to avoid over-foldings or self-intersections of the development. A brutal force approach is to enforce the requirements that all the possible triangulations of the

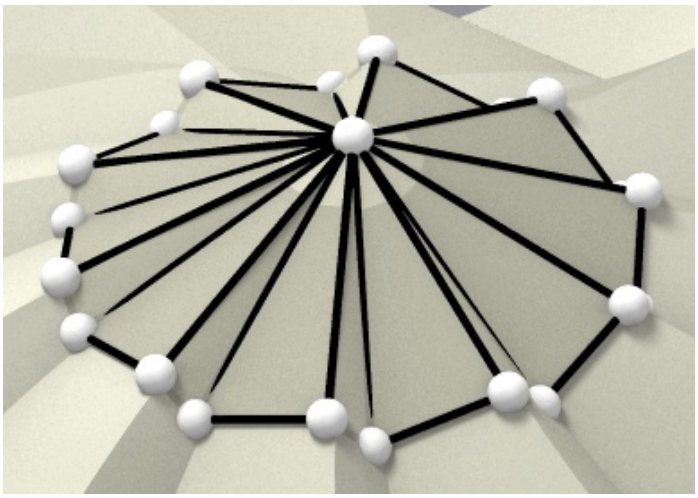


Figure 3.24: The developability condition of vertices where splines meet can be imposed strictly by the developability of the vertex star formed together with the nearby control points.

developed faces are consistently oriented by introducing signed areas as auxiliary variables. However, such an approach is not numerically robust, especially when the quads are too narrow due to over discretized parameters. Therefore, a superior alternative is to use the polyline fairness energy of the developed discretization.

3.4.3 Further examples

Examples of cylinder topology. The curved-crease sculpture presented in Figure 3.26 is inspired by David Huffman's *hexagonal column* design [DDK11]. This surface does not exhibit developability onto a plane, but onto a right circular cylinder of radius r. To maintain developability during modeling, the cylinder is cut open and unrolled onto the plane. The coordinates of vertices of the development $(\bar{V}, \bar{E}, \bar{F})$ are considered modulo the vector $(2\pi r, 0)$. Another cylindrical example is shown in Figure 3.27a.

Example: Flat Torus. Figure 3.27b shows a curved-crease sculpture of a torus topology. It has been initialized from a flat torus polyhedron, and has been mod-

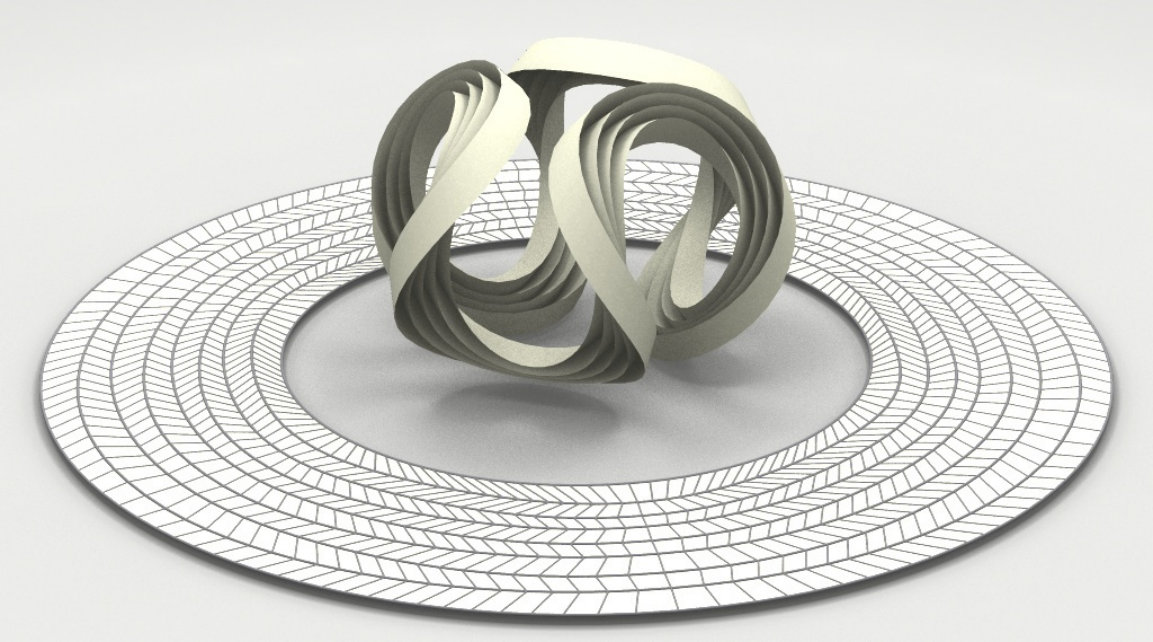


Figure 3.25: This curved-crease sculpture is folded from an annulus. It has been interactively created along the lines of Figure 3.20. The annulus unfolding shown here is the mesh $(\bar{V}, \bar{E}, \bar{F})$ used for internal computations. Its edges correspond to creases and rulings.

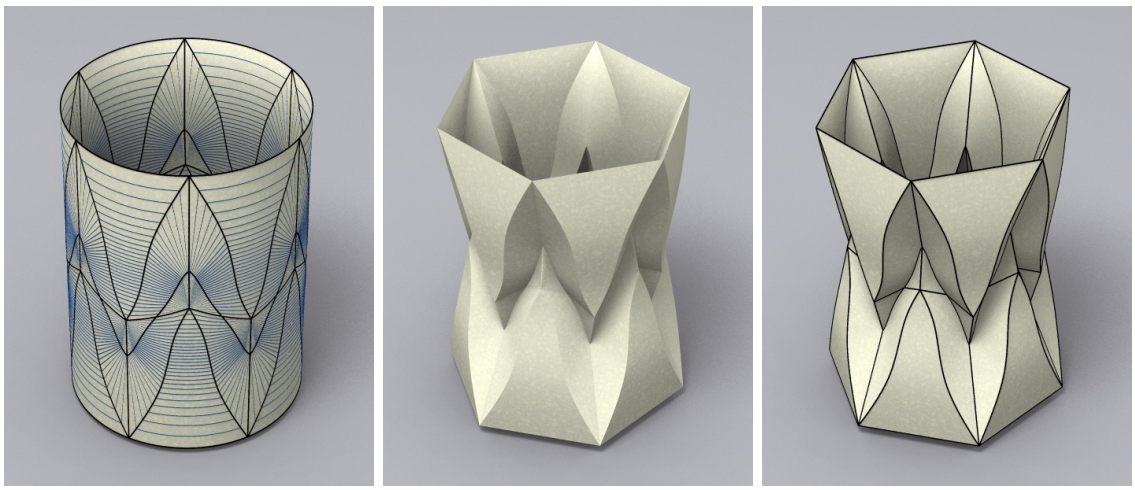


Figure 3.26: Interactive modeling of a cylindrical curved-crease sculpture. During modeling we maintain the property of unfoldability onto a cylinder (this is implemented via cutting the surface open and requiring a planar development with periodicity). *From left:* a user's initial sketch of the combinatorics, and the final sculpture with and without patch boundaries.

eled under constraints (3.2) and (3.10) which express local developability in faces and creases. As to vertices, the Gauss-Bonnet theorem says that the surface's total

curvature is zero. By symmetry, all vertices carry zero Gauss curvature.

3.5 Discussion

3.5.1 Choice of parameters

The choices of parameters are empirically guided by numerical experiments. Based on observations of tests with different parameters, the proposed approach is not highly sensitive with respect to the choice of parameters. The coefficient for the soft energy ϵ_1 influences the final shape while it has little affect on the achievable surface quality. The parameter for the closeness to previous iteration ϵ_2 can be chosen in the interval $[10^{-6}, 10^{-3}]$ (or $[10^{-6}, 10^{-2}]$, if curved-folding constraints are present) with only small changes in the final surface quality. The comparatively smaller values of ϵ_2 slightly improve the convergence rate, but larger values are usually preferred in order to achieve a better proximity to the initial shape. Frequently, ϵ_1 is set to zero after 5 iterations to improve convergence and to save time, since this regularizer has little

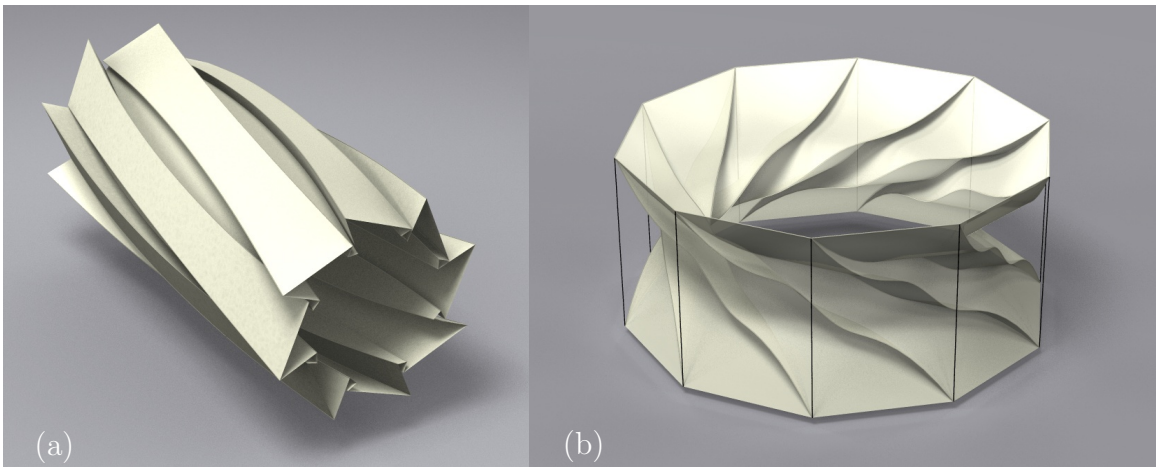


Figure 3.27: (a) Creases which enjoy the "geodesic" property can be simulated by three curved folds in close proximity. (b) This surface, whose outer faces are planar and have been removed for better visibility, has the topology of a torus and is intrisically flat.

effect in later stages of iteration. Generally, more constraints such as curved-folding conditions require higher values of ϵ_2 to prevent the shape drifting away from the initial. Some constraints such as maintaining prescribed development also have a regularizing effect. In these cases, ϵ_1 can be set to smaller values.

3.5.2 Limitation

A major limitation of this approach is that the decomposition combinatorics have to be specified before further computation. This limits the exploration of all possible configurations, as a pair of two isometric develoapble surfaces may no longer be transformable to each other computationally. This is also the restriction of many previous approaches, see e.g., [SVWG12]. A more flexible framework which allows automatic decomposition changes would still be desired and worth exploring in the future. However, it should also be noticed that having to choose combinatorics would not limit the possible shapes that could be modeled through the proposed approach.

Chapter 4

Further Extensions of *Guided Projection*

The previous chapters introduce *Guided Projection* with basic constraints. Examples include the design of polyhedral surfaces, self-supporting structures and developable surfaces including curved origami. The efficiency of *Guided Projection* that enables interactive editing is based on the insights that all the constraints should be formulated as quadratic or linear equations and solved together with soft energies serving as regularizers. The potential applications of *Guided Projection* are not limited to the scopes discussed above.

This chapter focuses on the extensions of *Guided Projection*, which could be roughly divided into two categories. On the one hand, the generalization of hard constraints requires the formulation of practical considerations to quadratic equations. Often, such formulations can only be achieved with the introduction of auxiliary variables. On the other hand, defining general guiding energies requires abstracting aesthetic or practical considerations to novel regularizers. It is important that the regularizers are meaningful in the sense that the regularized problems are neither over-regularized without feasible solutions nor under-regularized with too many

undesirable solutions.

In the following discussion, further formulations of hard constraints are exemplified through structures with repetitive elements, e.g., congruent nodes, or edges of the same length. Additionally, generalization of the regularizers is illustrated through *polyhedral patterns*, which are meshes with planar faces arranged regularly. To find the feasible regularity, explicit solutions of polyhedral patterns are constructed on paraboloids of positive, zero and negative Gaussian curvatures with different *strip decompositions*. The construction of explicit solutions serves as the foundation of *affine symmetry based regularizers*, guiding the computation of polyhedral patterns on freeform surfaces with a variety of arrangements.

4.1 Extensions of the hard constraints

Construction with repetitive elements reduces the manufacturing cost by enabling the reuse of molds or manufacturing settings. Such a goal could be easily achieved for geometrically simple traditional buildings featuring straight lines or planar facades. However, creating freeform surfaces with repetitive elements is highly intricate. For example, French architect Alain Lobel studied meshes with equilateral triangles[JTT+14]. These meshes are also named as Lobel frames, and can be constructed by the same beams. Alain Lobel took a constructive approach to generate some examples, but only special types of structures are created. With *Guided Projection*, the enforcement of repetitive geometric elements can be easily formulated and incorporated during computation[JWWP14, JTT+14].

Following the trend of biomimetic design and inspired by the amazing constructions of bees, Jiang and his colleagues study the design of *freeform honeycomb structures* [JWWP14]. In their work, the honeycomb structures are formed by connecting

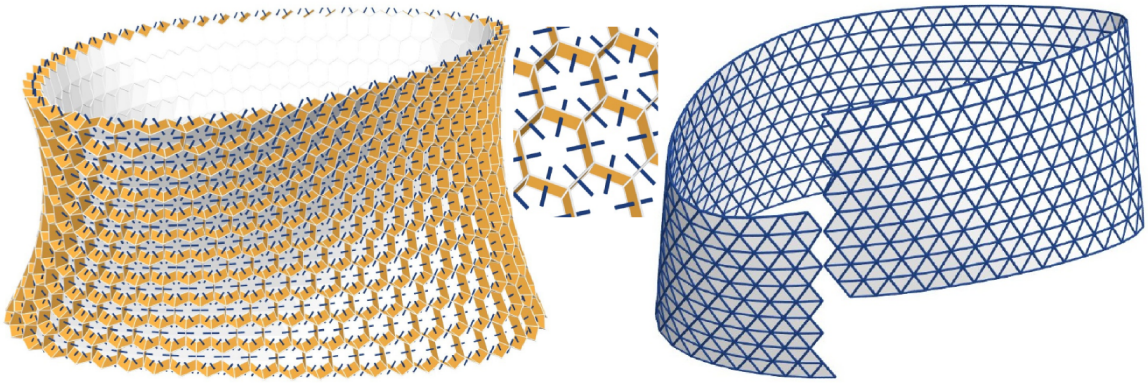


Figure 4.1: Enforcing structural similarity can create structures with repetitive elements such as the honeycomb structure with congruent nodes (left) and the *Lobel frame* with the same beams (right). They are related by a special duality condition (middle)[JTT+14].

two hex-dominant meshes which are roughly offsets of each other. Each pair of the corresponding edges is connected by a planar wall. Similar to the honeycomb structures in nature, the design of freeform honeycomb structures also requires the angles between adjacent walls be 120 degrees. Due to the congruency of the wall axis, all of the nodes can be batch manufactured based on extrusion and cutting, which greatly benefits the engineering process.

There exists an interesting duality between the freeform honeycomb structures and the Lobel frames[JTT+14, JTW+15]. Translating and then followed by connecting the wall normals of a freeform honeycomb structure creates a Lobel frame. Similarly, placing planes orthogonal to the edges of a Lobel frame forms a freeform honeycomb structure. Figure 4.1 presents examples.

4.2 Regularizers for polyhedral patterns

In addition to generalizing hard constraints, soft energies guiding the computation can also be extended. An interesting sceinario is finding the feasible regularity of

polyhedral patterns, which are meshes with planar faces arranged in regular ways [JTV+15] [1].

The main purpose of creating polyhedral patterns is to enrich the design space of polyhedral surfaces, from triangle or quad dominant meshes to those with various connectivities and polygons of higher valences. To keep the problem tractible, this study focuses on patterns with combinatorics equivalent to semi-regular tilings of a plane, where every vertex is congruent to each other.

To find the feasible regularities for creating general polyhedral patterns, this section focuses on explicit constructions of polyhedral patterns inscribed to paraboloids with positive, zero and negative Gaussian curvatures. Such explicit solutions are foundations for the proposed regularizers to guide the computation of polyhedral patterns approximating freeform surfaces.

The main questions here are: How should the faces be arranged and also how much geometric regularity a planar pattern can maintain when it is mapped onto a surface S so that the faces remain planar? Answers to these questions guide the formulation of the algorithm in various stages, especially the introduction of the regularizers based on symmetries.

4.2.1 Plane-paraboloid intersection

Each face of a polyhedral pattern, inscribed to the paraboloid $S_2 : 2z = \kappa_1 x^2 + \kappa_2 y^2$, $\kappa_1\kappa_2 \neq 0$, has its vertices restricted on the intersection of S_2 with a plane $z = ax + by + c$. The intersection curve is a conic in general, and its orthogonal projection onto the x, y-plane, or its top view has a form of $\kappa_1(x - x_0)^2 + \kappa_2(y - y_0)^2 = d$, where

[1] This project is a close collaboration with Caigui Jiang and other colleagues. The author's contribution is mainly on the theoretical side such as the explicit constructions discussed in this section.

$x_0 = \frac{a}{\kappa_1}$, $y_0 = \frac{b}{\kappa_2}$, and $d \neq 0$ if we exclude the case where the plane is tangent to the paraboloid for now.

The simplest is a rotational paraboloid, e.g. $S_2 : 2z = x^2 + y^2$, where the top views of the conics are circles (see Figure 4.2, left). For a regular or a semi-regular pattern in the x, y-plane, each of its faces is regular and thus inscribed to a circle. Lifting the vertices $(x_i, y_i, 0)$ of the pattern to points $(x_i, y_i, (x_i^2 + y_i^2)/2)$ on S_2, circles are lifted to ellipses and thus faces remain planar. We obtain a polyhedral pattern P with vertices on S_2 (see Figure 4.3, left).

For an elliptic paraboloid $\kappa_1\kappa_2 > 0$, the top views of the intersections are similar ellipses which can be mapped to each other by translation and uniform scaling. As an elliptic paraboloid is an affine image of a rotational paraboloid, mapping a planar pattern to it is also straightforward.

For a hyperbolic paraboloid $\kappa_1\kappa_2 < 0$ (Figure 4.2, right, and Figure 4.3, right), the top views of the intersections are hyperbolae with parallel asymptotes, which could be transformed to each other by translation, scaling and conjugacy, where a pair of

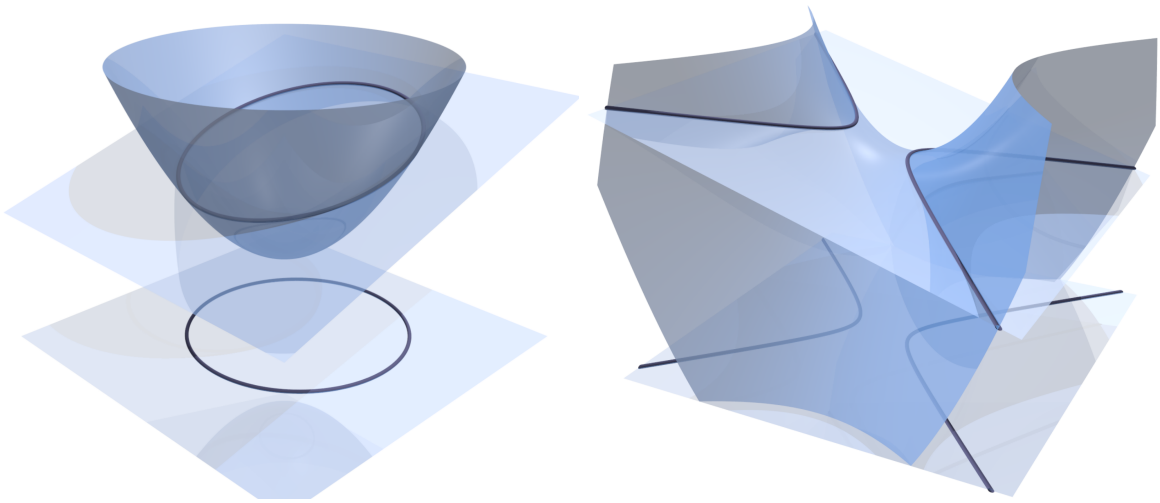


Figure 4.2: The orthogonal projection onto the x, y-plane of the intersection curve with a plane is a circle for a rotational paraboloid (left), an ellipse for an elliptic paraboloid, and a hyperbola for a hyperbolic paraboloid (right).

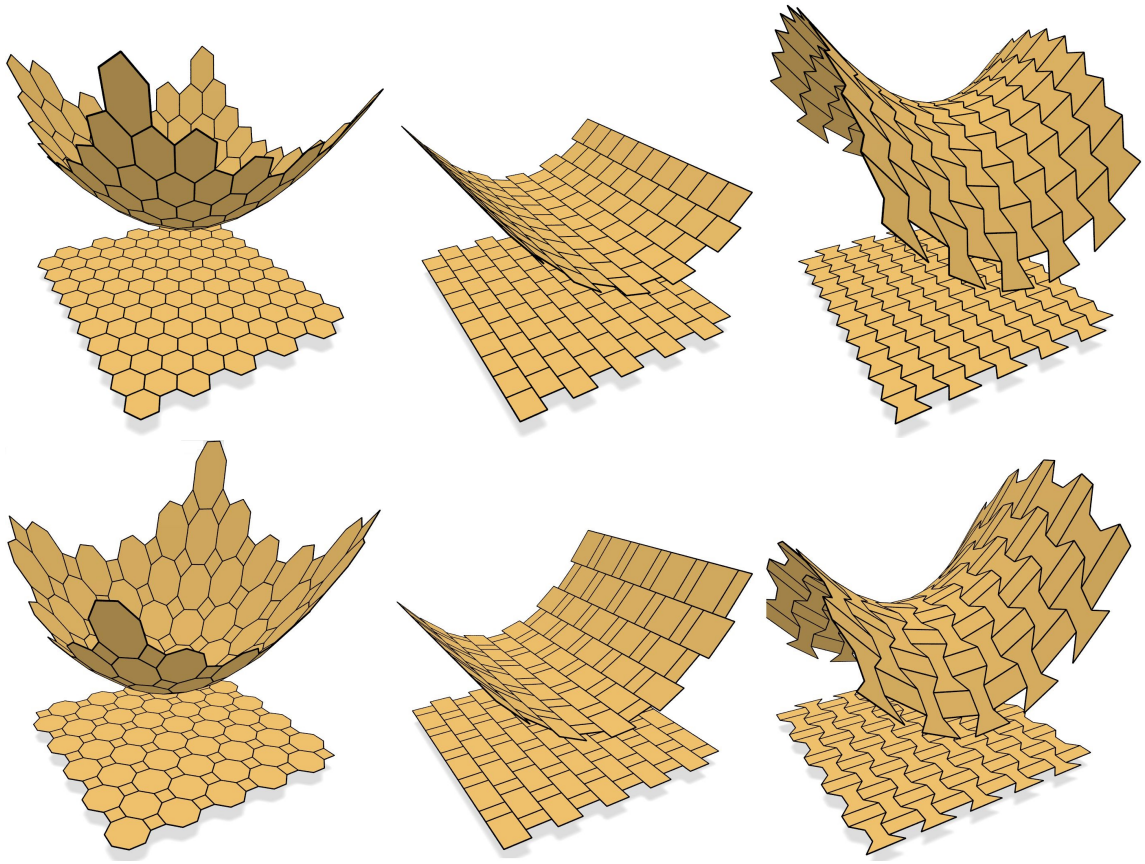


Figure 4.3: The transformation of polyhedral patterns from a rotational paraboloid (left) via a parabolic cylinder (middle) to a hyperbolic paraboloid (right).

conjugate hyperbolae are $\kappa_1(x-x_0)^2+\kappa_2(y-y_0)^2=\pm d$.

Conjugacy and metric The curvatures of the paraboloid also define a quadratic form $<\mathbf{p},\mathbf{q}>:=\kappa_1 x_p x_q+\kappa_2 y_p y_q$ in the x,y-plane. Two directions $\mathbf{p}$ and $\mathbf{q}$ are *conjugate* if $<\mathbf{p},\mathbf{q}>=0$, which can be regarded as the orthogonality condition with this quadrtic form. The quadratic form also induces a metric $\|\mathbf{p}\|^2:=<\mathbf{p},\mathbf{p}>$, and all the conics can be regarded as circles under such a metric. When $\kappa_1\kappa_2>0$, the metric is essentially Euclidean. When $\kappa_1\kappa_2<0$, the metric is *pseudo-Euclidean*.

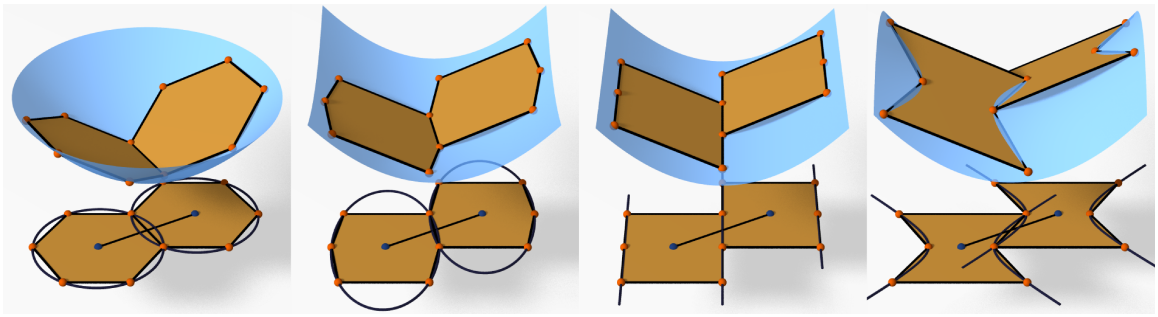


Figure 4.4: On the top view, a primal edge shared by two faces is a common chord shared by two circumconics, which is conjugate to and bisected by the dual edge, defined as the connecting line of the circumconic centers.

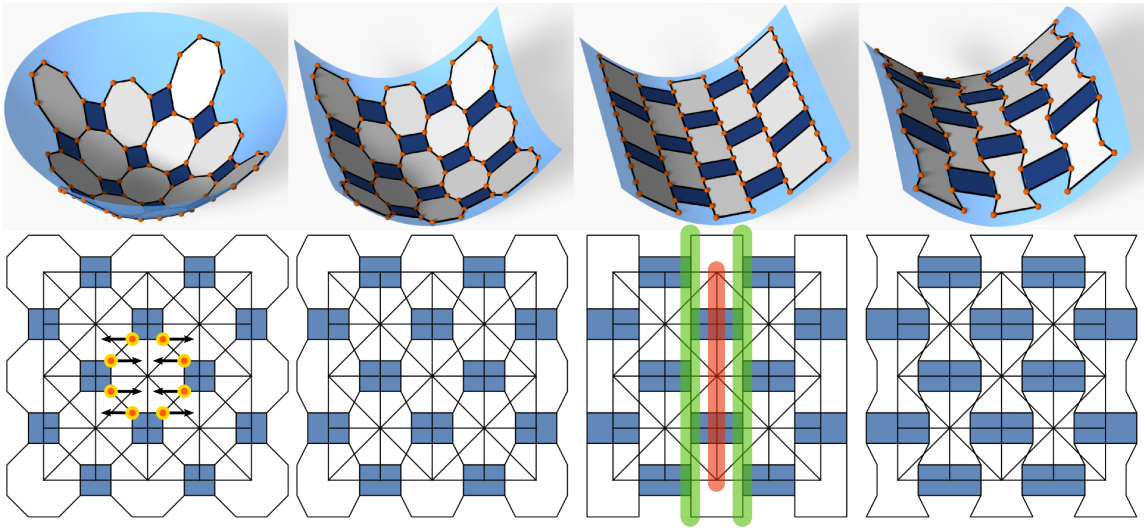


Figure 4.5: Deforming and lifting. Top: the paraboloid tilings. Bottom: top-views of the tilings. Left to right: original (fit to the canonical rotational paraboloid), elliptic, cylindrical, and hyperbolic.

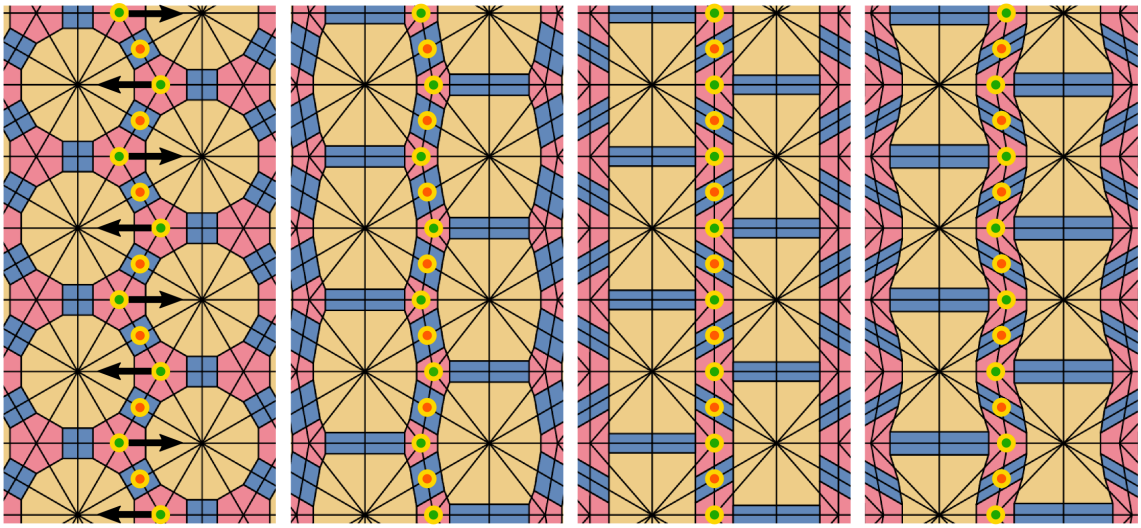


Figure 4.6: Pairs of primal and dual $(4, 6, 12)$ patterns on positive, cylindrical and negative curvature regions. Both the primal and the duals are alighed with the ruling for the cylindrical paraboloid.

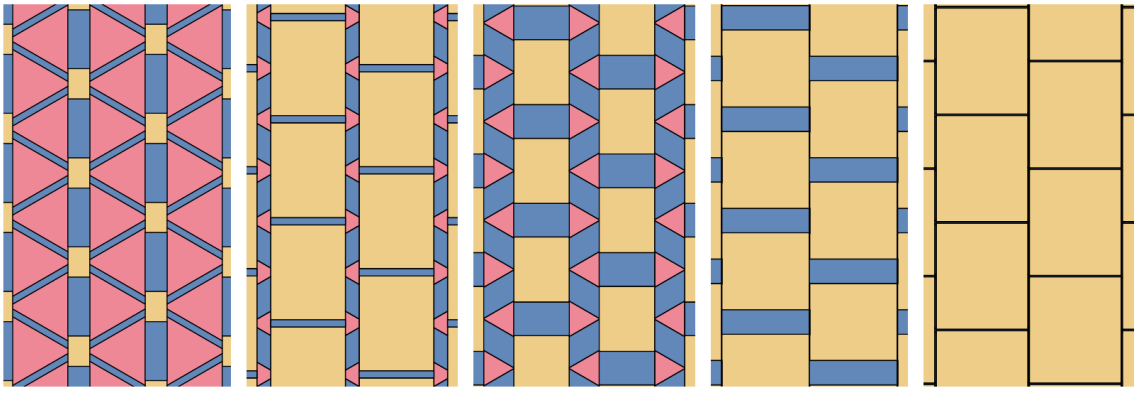


Figure 4.7: The primal $(4, 6, 12)$ patterns have 2 degrees of freedom when a feasible dual pattern is provided.

Consistency For a pattern that could be lifted to a paraboloid while keeping faces planar, the initial polygons are inscribed to similar or conjugate conics. Connecting conic centers of adjacent faces of a liftable *primal pattern* gives a *dual pattern*. The dual edges are conjugate to and bisected by the primal edges, which are called the consistency conditions, as illustrated in Figure 4.4 and Figure 4.5. Similarly, when there exists a pair of primal and dual patterns satisfying the consistency conditions, the primal is liftable to paraboloids without sacrificing face planarity. Figure 4.6 and Figure 4.7 presents examples. The proofs are provided in the appendix.

4.2.2 Strip decompositions

Parabolic cylinders It is interesting and insightful to observe the invariant regularity of the polyhedral patterns when the Gaussian curvatures of the paraboloids change from positive to negative. For that, one simple way is to set $\kappa_1 = 1$ and let κ_2 vary from 1 via 0 to -1; see Figure 4.3. The parabolic cylinder $2z = x^2$ is extremely important as it can be viewed as the limit and transitional state of elliptic and hyperbolic paraboloids.

The top view of an intersection curve of a parabolic cylinder with a plane is either a pair of parallel lines or a parabola. This study focuses on the former.

Strip decompositions The assignment of polygons on a parabolic cylinder is a combinatorial problem which we call *strip decomposition*. Pairs of lines are shared by sequences of polygons without dangling vertices in between, consequently, a feasible strip decomposition should satisfy these conditions:

1. The adjacent faces of a vertex do not all belong to a same strip.

2. Every inner face shares a common edge with two faces on the same strip.

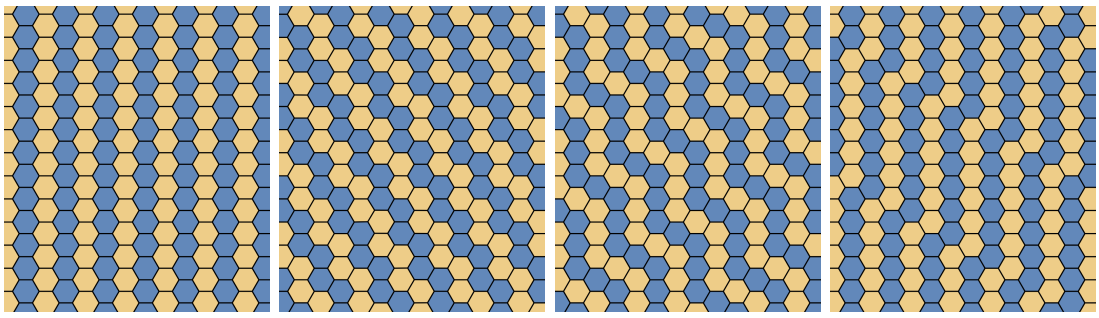


Figure 4.8: Different strip decompositions for regular hexagons. The corresponding polyhedral patterns of the two on the left are shown in 4.3, and the second pattern from the right is shown in 4.9.

A strip decomposition of a polyhedral pattern implies an assignment of vertices to rulings for the dual pattern. Therefore, if a pattern can be decomposed to strips, the same is true for its dual pattern. Regular quads, hexagons, $(3^4, 6)$, $(3, 4, 6, 4)$, $(4, 6, 12)$, $(4, 8, 8)$ [2] as well as their dual patterns have an infinite number of strip decompositions. We show two different strip decompositions for the regular hexagon pattern and the pattern (4, 8, 8) each in Figure 4.8. However, patterns $(3, 6, 3, 6)$, $(3, 12, 12)$ and their duals cannot be decomposed to strips.

For the $(3, 6, 3, 6)$ pattern, starting from any hexagon, an arbitrary triangle could be chosen first. The second condition requires the next choice to be another hexagon. Due to the first condition, the other triangle adjacent to the two chosen hexagons cannot be picked. However, this triangle cannot be assigned to another strip neither, as it cannot satisfy the second requirement with only one free edge left. Similar arguments are true for any patterns with triangles sharing no edges with other triangles, e.g., (3,12,12) patterns.

Symmetries of strip decompositions For a given pattern, a different choice of strip decompositions implies different regularity. For each strip decomposition, it is

[2]Naming of the patterns follows the convention according to the valences of the adjacent faces of each vertex. For example, a $(6, 6, 6)$ pattern is a hexagonal mesh, and a $(3, 4, 6, 4)$ pattern has triangles, quadrilaterals and hexagons.

interesting to see which symmetries a polyhedral pattern can keep when the parabolic cylinder is deformed to a hyperbolic or elliptic paraboloid.

It is helpful to examine the regular hexagon patterns, starting with the most straight forward strip decomposition shown on the left of Figure 4.8. Such a decomposition corresponds to a brick wall pattern when it is decomposed on a parabolic cylinder, as shown in Figure 4.3. On the top view, every hexagon is symmetric to the center and congruent to each other, and every pair of adjacent faces is symmetric with respect to the midpoint of their common edge.

When the pattern is lifted to the surface, the faces are also symmetric with respect to their barycenters. Though the faces are no longer congruent to each other, they can be transformed to each other by affine transformations. Every pair of adjacent faces have *affine symmetry with respect to an axis*, i.e., they could be mapped to each other through an affine map which keeps the axis fixed. In this case, the axis is the vertical line passing through the midpoint of their common edge. Of significance, with a given strip decomposition of a polyhedral pattern, the affine symmetries exist not only on the parabolic cylinders, but also on elliptic and hyperbolic paraboloids. The affine symmetries can also be preserved under affine transformations of the surfaces.

With the strip decomposition shown on the mid-left of Figure 4.8, the top views of the hexagons on a parabolic cylinder are isosceles trapezoids; see Figure 4.9. Instead of symmetries with respect to their centers, the faces have Euclidean reflective symmetries on the top view, and *affine (reflective) symmetries with respect to a plane and a direction* on the surface. Here, not all pairs of adjacent faces have affine symmetries any longer, but those that are symmetric on the parabolic cylinder do preserve the affine symmetries on paraboloids.

Similar analyses could be done for other strip decompositions and other patterns. In general, *the available symmetries of polyhedral patterns are revealed from the strip*

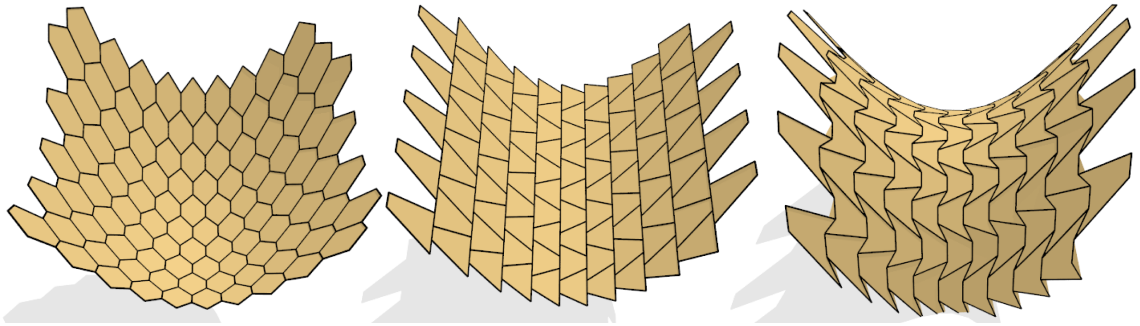


Figure 4.9: Transformation of the regular hexagon pattern from a rotational paraboloid (left) via a parabolic cylinder (middle) to a hyperbolic paraboloid (right) with the strip decomposition shown in Figure 4.8, the second from right.

decompositions.

4.2.3 Connections to regularizers

The constructed polyhedral patterns inscribed to paraboloids do not exhibit the encoded properties of commonly used regularizers such as angle bounds, polyline fairness, Laplacian smoothness. Therefore, when planarity is required, these regularziers lead to over-constrained problems and cannot provide sufficient flexibility.

The polyhedral patterns could preserve many affine symmetries locally when approximating paraboloids with a given strip decomposition, thus motivating the use of symmetries as regularizers for general surfaces. For each pattern, affine symmetries could be specified and assigned according to the choice of a valid strip decomposition. To approximate general surfaces together with the planarity requirements, the assigned symmetries provide sufficient flexibility and regularization for the patterns. A computational result on a freeform surface is demonstrated in Figure 4.10. Results with different strip decomposition are illustrated in Figure 4.11.

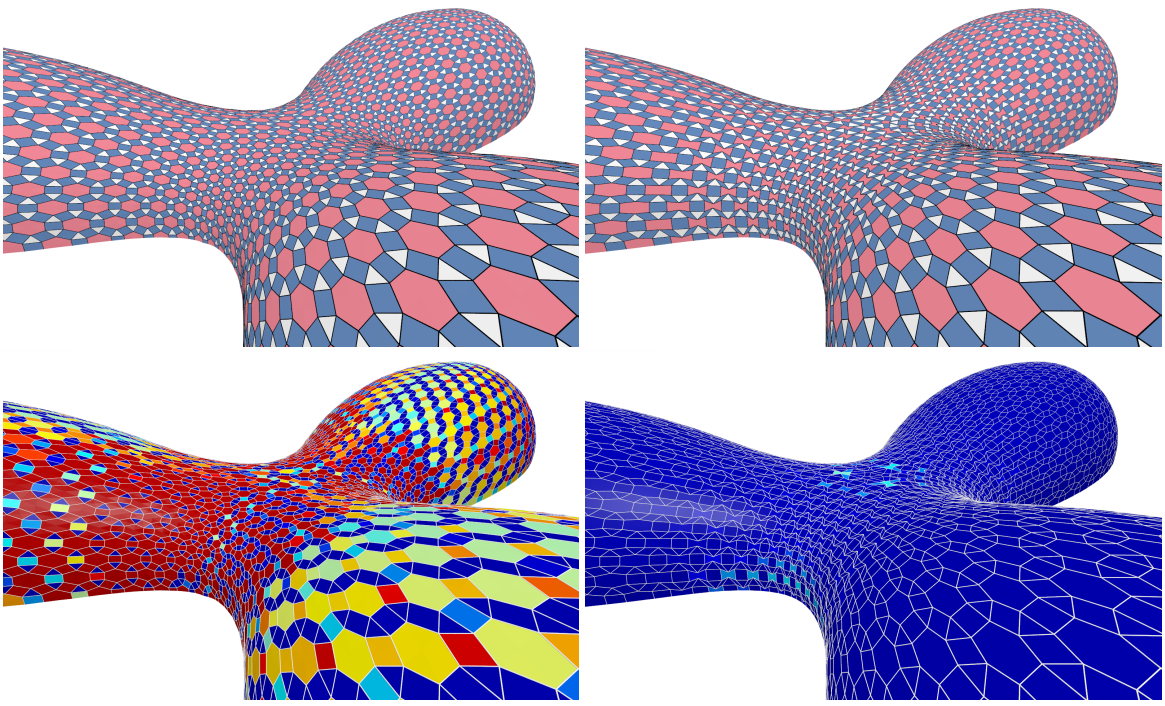


Figure 4.10: With symmetry based regularizers, the non-planar pattern shown on the left is planarized towards a polyhedral pattern shown on the right, with planar faces arranged regularly.

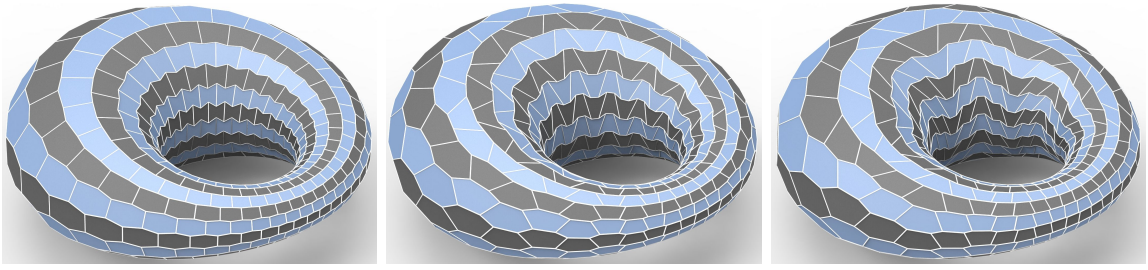


Figure 4.11: Different strip decompositions increase the available types of polyhedral patterns. With the three strip decompositions shown in Figure 4.8, we obtain hexagon patterns approximating a cyclide with different appearances.

Chapter 5

Conclusion and Outlook

The previous chapters of this thesis introduce the basic approach of *Guided Projection* and several extensions. Those formulations and applications have also opened many new directions and challenges. For future work, theoretical investigations are vital sources for intuitions and insights for computation. In the meanwhile, computational approaches are best studied with concrete examples, as the challenges from real applications provide interesting research questions. To conclude the thesis, this chapter samples a few possible directions to explore in future to further extend the content presented in this thesis.

5.1 Theoretical investigations

5.1.1 Discrete differential geometry

Discrete differential geometry is a field which can provide many insights for possible geometric solutions. For example, explicit constructions of polyhedral patterns mentioned in the previous chapter have led to the new regularity notion based on symmetries. Many topics in discrete differential geometry, including regularity, rigidity, stability, integrability, transformation, subdivision and convergence of different

geometric structures could provide further foundations for general approaches of computational design.

5.1.2 Volumetric computation

Beyond computation with surfaces, trending manufacturing techniques like 3D printing, CNC machining and functional assembling require volumetric representations. Though this thesis mainly focuses on surfaces, *Guided Projection* can be naturally generalized to volumes, with the control of enclosed volumes demonstrated above being a simple example. At the same time, volumetric computation implies an increased number of variables. Seeking proper sparse representations and devising computational approaches with reduced dimensionality is important for efficiency.

5.1.3 Combinatorial optimization

Most computational design platforms demonstrated in this thesis allow users to specify or change the connectivities before or during the interactive design phase. In the meanwhile, providing automatic suggestions of connectivity changes is often desired and left for future work. Besides mesh connectivities, combinatorial problems are omnipresent, like the assembly of functional parts, and enforcing repetivity with multiple types of elements. Those challenges share seemingly similarities, however, different computational tools are required to be advanced for suitable solutions.

5.1.4 Multi-resolutional approaches

Design with subdivision or control structures has been widely used in the projects of this thesis for intuitive and stable control of the shape and connectivity. There is a huge potential to accelerate computational speed by leveraging multi-resolutional

structures. Such approaches could enable interactivity for more complicated structures, where manipulation and computation on coarser levels can be interactively reflected in finer scales, and vice versa.

5.1.5 Learning based computational design

Existing designs are rich sources of inspirations for current designers, especially those evolved over a long time span. Many of the design decisions are made based on practical experiences or personal tastes that may not have a clear mathematical formulation. It would be interesting and effective to incorporate learning based techniques into the development of future computational methodologies for design with a particular style of an artist or the tradition of a school.

5.1.6 Discrete physical models

In the project of form-finding with polyhedral meshes, the static equilibrium formulations based on simplified physical models lead to computational results which are verifiable by commercial simulation packages. Incorporating simple yet efficient physical models into computation requires insights to both the physics and the design scenario. Similar to the philosophy of discrete differential geometry, discretizing physical models and theories, instead of the equations, is necessary for efficient computational design with faithful physical assumptions.

5.2 Broader applications

5.2.1 Folding patterns and transformable structures

A natural extension based on both developable surfaces and polyhedral patterns are origami tessellation patterns. Straight folds enable deployable structures that could be compactly compressed and transported. The more challenging curved origami tessellations allow more expressive artistic creations. A subdivision based scheme with controlled roughness to create folding patterns would be quite interesting to look into. At the same time, general computational methodologies for transformable structures are also desirable.

5.2.2 Stability

Static equilibrium does not imply stability under external perturbations. For safety and robustness, it is important to study static stabilities of mechanical structures and architectural designs, for both the construction sequences as well as the final products. Beyond static stability, structures with dynamic stabilities like rattlebacks and more general motorized systems are also fascinating topics.

5.2.3 Details and textures

A common approach to achieve interactive rates is to deal with less variables through computation with coarse representations, which often compromises the details. There is a huge potential to expedite the computational speed for the design of large and complex structures with fine details or precise parts through multi-resolutional and procedural approaches. With additional controlled stochasticity, this problem also shares great similarities with texture synthesis.

5.2.4 Design retrieval and history of designs

Architects, engineers, archaeologists and designers are interested to study how designs evolve over years or centuries by different people in varied geographic regions. It would be very helpful for them to have tools to retrieve historical designs, and to compare the shapes, assemblies and functionalities of them, in a way similar to taking time-lapse photos of designs, to find deeper stories behind the evolution.

5.2.5 Metamaterials

Existing physical models empowered computational tools to simulate materials as forward problems. Design of materials for novel electromagnetic or mechanical properties have led to very interesting inverse problems with many open questions. Computational design of metamaterials, such as those with negative Poisson ratios, could be natural extensions of computational designs towards one extreme of scales.

REFERENCES

[Aum91] Gunter Aumann. Interpolation with developable Bézier patches. *Comput. Aided Geom. Des.*, 8:409–420, 1991.

[Aum03] Gunter Aumann. A simple algorithm for designing developable Bézier surfaces. *Comput. Aided Geom. Des.*, 20:601–619, 2003.

[Blo09] Philippe Block. *Thrust Network Analysis: Exploring Three-dimensional Equilibrium*. PhD thesis, M.I.T., 2009.

[BML+14] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4):#152, 1–11, 2014. Proc. SIGGRAPH.

[BO07] Phillipe Block and John Ochsendorf. Thrust network analysis: A new methodology for three-dimensional equilibrium. *J. Int. Assoc. Shell and Spatial Structures*, 48(3):167–173, 2007.

[BPC+10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

[BSWP12] Sofien Bouaziz, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. Shape-up: Shaping discrete geometry with projections. *Comp. Graph. Forum*, 31:1657–1667, 2012. Proc. SGP.

[CC04] Chih-Hsing Chu and Jang-Ting Chen. Geometric design of uniform developable B-spline surfaces. In *Proc. DETC*, volume 1, pages 431–436. 2004.

[CS02] Chih-Hsing Chu and Carlo Séquin. Developable Bézier patches: properties and design. *Comput. Aided Des.*, 34:511–527, 2002.

[dB78] Carl de Boor. *A Practical Guide to Splines*. Springer, 1978.

[DBD+13] Bailin Deng, Sofien Bouaziz, Mario Deuss, Juyong Zhang, Yuliy Schwartzburg, and Mark Pauly. Exploring local modifications for constrained meshes. *Comp. Graph. Forum*, 32(2):11–20, 2013. Proc. Eurographics.

[DDH+11] Erik Demaine, Martin Demaine, Vi Hart, Gregory Price, and Tomohiro Tachi. (Non)existence of pleated folds: how paper folds between creases. *Graphs and Combinatorics*, 27:377–397, 2011.

[DDK11] Erik Demaine, Martin Demaine, and Duks Koschitz. Reconstructing David Huffman's legacy in curved-crease folding. In *Origami 5*, pages 39–52. A. K. Peters, 2011.

[DDMS12] Marcelo Dias, Levi Dudte, L. Mahadevan, and Christian Santangelo. Geometric mechanics of curved crease origami. *Phys. Rev. Lett.*, 109(114301):1–13, 2012.

[dGAOD13] Fernando de Goes, Pierre Alliez, Houman Owhadi, and Mathieu Desbrun. On the equilibrium of simplicial masonry structures. *ACM Trans. Graph.*, 32(4):#93, 1–10, 2013. Proc. SIGGRAPH.

[EKS+10] Michael Eigensatz, Martin Kilian, Alexander Schiftner, Niloy J Mitra, Helmut Pottmann, and Mark Pauly. Paneling architectural freeform surfaces. volume 29, pages #45, 1–10, 2010. Proc. SIGGRAPH.

[Huf76] D. A. Huffman. Curvature and creases: A primer on paper. *IEEE Trans. Computers*, 25:1010–1019, 1976.

[JTT+14] Caigui Jiang, Chengcheng Tang, Marko Tomičić, Johannes Wallner, and Helmut Pottmann. Interactive modeling of architectural freeform structures – combining geometry with fabrication and statics. In Philippe Block et al., editors, *Advances in Architectural Geometry 2014*. Springer, 2014.

[JTV+15] Caigui Jiang, Chengcheng Tang, Amir Vaxman, Peter Wonka, and Helmut Pottmann. Polyhedral patterns. *ACM Trans. Graph.*, 34(6):#172, 1–12, 2015. Proc. SIGGRAPH Asia.

[JTW+15] Caigui Jiang, Chengcheng Tang, Jun Wang, Johannes Wallner, and Helmut Pottmann. Freeform honeycomb structures and lobel frames. In *ACM SIGGRAPH 2015 Posters*, page 75. ACM, 2015.

[JWWP14] Caigui Jiang, Jun Wang, Johannes Wallner, and Helmut Pottmann. Freeform honeycomb structures. *Comput. Graph. Forum*, 33(5):185–194, 2014.

[KFC+08] Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy Mitra, Alla Sheffer, and Helmut Pottmann. Curved folding. *ACM Trans. Graph.*, 27(3):# 75, 1–9, 2008. Proc. SIGGRAPH.

[Liu10] Yang Liu. HLBFGS. Software available from *http://research.microsoft.com/en-us/UM/people/yangliu/software/HLBFGS/*, 2010.

[LPS+13] Yang Liu, Hao Pan, John Snyder, Wenping Wang, and Baining Guo. Computing self-supporting surfaces by regular triangulation. *ACM Trans. Graph.*, 32(4):#92, 1–10, 2013. Proc. SIGGRAPH.

[LPW+06] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.*, 25(3):681–689, 2006. Proc. SIGGRAPH.

[LR92] Johann Lang and Otto Röschel. Developable $(1, n)$-Bézier surfaces. *Comput. Aided Geom. Des.*, 9:291–298, 1992.

[LXW+11] Yang Liu, Weiwei Xu, Jun Wang, Lifeng Zhu, Baining Guo, Falai Chen, and Guoping Wang. General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph.*, 30(6):#140, 1–10, 2011. Proc. SIGGRAPH Asia.

[Max64] J.C. Maxwell. On reciprocal diagrams and diagrams of forces. *Philosophical Magazine*, 4(27):250–261, 1864.

[MC98] T. Maekawa and J. Chalfant. Design and tesselation of B-spline developable surfaces. *J. Mech. Des.*, 120:453–461, 1998.

[MI11] Jun Mitani and Takeo Igarashi. Interactive design of planar curved folding by reflection. In *Pacific Graphics, Short Papers*, pages 77–81. 2011.

[Mit12] Jun Mitani. Origami applications. *http://mitani.cs.tsukuba.ac.jp/origami_application*, 2012.

[MK12] Neil Meredith and James Kotronis. Self-detailing and self-documenting systems for wood fabrication: The Burj Khalifa. In *Advances in Architectural Geometry 2012*, pages 185–198. Springer, 2012.

[ML12] Meher McArthur and Robert J. Lang. *Folding Paper: The infinite possibilities of Origami*. Int. Arts & Artists, Wash. DC, 2012.

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

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

[PF95] Helmut Pottmann and Gerald Farin. Developable rational bézier and b-spline surfaces. *Comput. Aided Geom. Des.*, 12(5):513–531, 1995.

[POG13] Roi Poranne, Elena Ovreiu, and Craig Gotsman. Interactive planarization and optimization of 3D meshes. *Comp. Graph. Forum*, 32(1):152–163, 2013.

[PSB+08] H. Pottmann, A. Schiftner, P. Bo, H. Schmiedhofer, W. Wang, N. Baldassini, and J. Wallner. Freeform surfaces from single curved panels. *ACM Trans. Graph.*, 27(3):#76, 1–10, 2008. Proc. SIGGRAPH.

[SLB+12] Alexander Schiftner, Nicolas Leduc, Philippe Bompas, Niccolo Baldassini, and Michael Eigensatz. Architectural geometry from research to practice — the Eiffel Tower Pavilions. In *Advances in Architectural Geometry 2012*, pages 213–228. Springer, 2012.

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

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

[TBWP] Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. Interactive design with developable surfaces. *ACM Trans. Graph.* to appear.

[TM12] Tomohiro Tachi and Koryo Miura. Rigid-foldable cylinders and cells. *J. Int. Assoc. Shell & Spatial Structures*, 53(4):217–226, 2012.

[TSG+14] Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. Form-finding with polyhedral meshes made simple. *ACM Trans. Graph.*, 33(4):#70, 1–9, 2014. Proc. SIGGRAPH.

[VHWP12] Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. Design of self-supporting surfaces. *ACM Trans. Graph.*, 31(4):#87, 1–11, 2012. Proc. SIGGRAPH.

[WC11] K Wang and Y Chen. Folding a patterned cylinder by rigid origami. In *Origami 5*, pages 265–276. A.K. Peters, 2011.

[YYPM11] Yongliang Yang, Yijun Yang, Helmut Pottmann, and Niloy Mitra. Shape space exploration of constrained meshes. *ACM Trans. Graph.*, 30(6):#124,1–11, 2011. Proc. SIGGRAPH Asia.

[ZTY+12] Xin Zhao, Chengcheng Tang, Yongliang Yang, Helmut Pottmann, and Mitra Niloy. Intuitive design exploration of constrained meshes. In *Advances in Architectural Geometry 2012*, pages 305–318. 2012.

APPENDICES

Appendix A

Consistency of Explicitly Constructed Polyhedral Patterns

We prove that a primal pattern can be vertically lifted to a paraboloid S, while keeping every face planar, if and only if every primal edge in the pattern is conjugated to, and bisected by the corresponding dual edge (i.e., the edge between the conic centers of the neighboring faces). The paraboloid S is defined as $z = \kappa_1 x^2 + \kappa_2 y^2$. We currently assume $\kappa_1 \kappa_2 \neq 0$, and refer to this assumption further on. We consider the induced quadratic form $\langle \mathbf{a}, \mathbf{b} \rangle := \kappa_1 x_a x_b + \kappa_2 y_a y_b$ on the (x, y)-plane, and denote the squared norm $|\mathbf{a}|^2 = \langle \mathbf{a}, \mathbf{a} \rangle$ accordingly. The conjugacy relation is thus $\langle \mathbf{a}, \mathbf{b} \rangle = 0$. We prove our claim using the following:

LEMMA 1. *A polygon with vertices* $(\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ *on the* x, y*-plane, inscribed to a conic of the form* $|\mathbf{p} - \mathbf{c}|^2 = \gamma,\ \gamma \in \mathbb{R}$*, remains planar when lifted to* S*.*

Proof. The lifted points of $\mathbf{p}_i$ on the paraboloid are on the intersection of S of the vertical cylinder C extruding the conic: $\kappa_1(x - x_c)^2 + \kappa_2(y - y_c)^2 = \gamma$. Thus, they also lie on any linear combination of C and S. The surface $C - S$ is a plane, as the quadratic terms cancel out. □

LEMMA 2. *Let* $\mathbf{p}_j - \mathbf{p}_i$ *be a primal edge conjugate to the dual edge* $\mathbf{c}_j - \mathbf{c}_i$ *and bisected by* $\mathbf{c}_j - \mathbf{c}_i$ *at point* $\mathbf{d}$. *Then,* $|\mathbf{p}_i - \mathbf{c}_i|^2 = |\mathbf{p}_j - \mathbf{c}_i|^2$ *and* $|\mathbf{p}_i - \mathbf{c}_j|^2 = |\mathbf{p}_j - \mathbf{c}_j|^2$.

Proof. The conjugacy of $\mathbf{p}_j - \mathbf{p}_i$ and $\mathbf{c}_j - \mathbf{c}_i$ implies $\langle \mathbf{p}_j - \mathbf{p}_i, \mathbf{c}_j - \mathbf{c}_i \rangle = 0$, and the bisecting condition implies $|\mathbf{p}_i - \mathbf{d}|^2 = |\mathbf{p}_j - \mathbf{d}|^2$. Thus:

$$\begin{aligned} |\mathbf{p}_i - \mathbf{c}_i|^2 &= |\mathbf{d} - \mathbf{c}_i + \mathbf{p}_i - \mathbf{d}|^2 \\ &= \langle \mathbf{d} - \mathbf{c}_i, \mathbf{d} - \mathbf{c}_i \rangle + \langle \mathbf{p}_j - \mathbf{d}, \mathbf{p}_j - \mathbf{d} \rangle \\ &= \langle \mathbf{p}_j - \mathbf{c}_i, \mathbf{p}_j - \mathbf{c}_i \rangle. \end{aligned}$$

Similarly, $\langle \mathbf{p}_i - \mathbf{c}_j, \mathbf{p}_i - \mathbf{c}_j \rangle = \langle \mathbf{p}_j - \mathbf{c}_j, \mathbf{p}_j - \mathbf{c}_j \rangle$. □

COROLLARY 1. ***Consistency to planar lifting:*** *A primal pattern on the* (x, y)*-plane with every edge conjugate to and bisected by the dual edge, is liftable to* S *while keeping faces planar.*

Proof. Due to Lemma 2, for a given polygon $\{\mathbf{p}_i\}$ and center $\mathbf{c}$, we have $\langle \mathbf{p}_i - \mathbf{c}, \mathbf{p}_i - \mathbf{c} \rangle = \gamma$ for some constant $\gamma \in \mathbb{R}$, and thus the vertices are on a planar conic, and can be lifted to S while keeping face planarity, due to Lemma 1. □

We next prove how planar lifting leads to consistency.

LEMMA 3. *The vertical projection of the intersection curve between* S *and a plane* $P := z = ax + by + e$ *onto the* (x, y)*-plane, is a conic of the form* $|\mathbf{p} - \mathbf{c}|^2 = \gamma,\ \gamma \in \mathbb{R}$.

Proof. The surface obtained by the subtraction $P - S$ is a vertical cylinder passing through the intersection curve, which intersects the (x, y)-plane with a conic of the form $\kappa_1(x_p - x_c)^2 + \kappa_2(y_p - y_c)^2 = |\mathbf{p} - \mathbf{c}|^2 = \gamma$, where $x_c = \frac{a}{2\kappa_1}$, $y_c = \frac{b}{2\kappa_2}$, and $\gamma \in \mathbb{R}$. □

LEMMA 4. *On the (x, y)-plane, if two similar conics $c_i : |\mathbf{p} - \mathbf{c}_i|^2 = \gamma_i$ and $c_j : |\mathbf{p} - \mathbf{c}_j|^2 = \gamma_j$ intersect at $\mathbf{p}_i$ and $\mathbf{p}_j$, then $\mathbf{p}_j - \mathbf{p}_i$ is conjugate to and bisected by $\mathbf{c}_j - \mathbf{c}_i$.*

Proof. The subtraction of the two conics $c_j - c_i$ produces a line l defined by $\langle \mathbf{p}, \mathbf{c}_j - \mathbf{c}_i \rangle = const.$ As both $\mathbf{p}_i$ and $\mathbf{p}_j$ are on l, $\langle \mathbf{p}_j - \mathbf{p}_i, \mathbf{c}_j - \mathbf{c}_i \rangle = 0$.

To show that bisection holds as well, we apply a shearing transformation so that $\bar{\mathbf{c}}_j - \bar{\mathbf{c}}_i$ and $\bar{\mathbf{p}}_j - \bar{\mathbf{p}}_i$ become orthogonal, and thus aligned with the axes. As $\bar{c}_i$ and $\bar{c}_j$ are both reflectively symmetric with respect to $\bar{\mathbf{c}}_j - \bar{\mathbf{c}}_i$, so are their intersections. Therefore, the midpoint of $\mathbf{p}_j - \mathbf{p}_i$ lies on $\mathbf{c}_j - \mathbf{c}_i$ before the shearing transformation. □

COROLLARY 2. ***Planar lifting to consistency:*** *Consider a pattern on the (x, y)-plane which is the vertical projection of a polyhedral pattern inscribed on S. Then, there exists a dual pattern with each edge conjugate to and bisecting the corresponding primal edge.*

Proof. Due to Lemma 3, the faces of primal pattern are all inscribed to conics with the form of $|\mathbf{p} - \mathbf{c}_i|^2 = \gamma_i,\ \gamma_i \in \mathbb{R}$. Consider the dual pattern formed by connecting the adjacent face circumconic centers. Then, the primal edges $\mathbf{p}_j - \mathbf{p}_i$ are conjugate to and bisected by the dual edges $\mathbf{c}_j - \mathbf{c}_i$. □

Appendix B

Symmetries of PQ Patterns

Polyline fairness is a popular choice for a regularizer for planar-quadrilateral meshes in the literature. The objective is to straighten the polylines by minimizing second-order differences between neighboring vertices on a polyline, and this is equivalent to applying symmetries with respect to vertices. However, this fairness measure is restrictive in the sense that it is only applicable for meshes which are aligned with conjugate directions on the reference surface. Our computation with edge-midpoint symmetries introduces more degrees of freedom, and is less sensitive to the initial alignment.

In the following, we explain why the edge-midpoint symmetry regularizer is more flexible than the polyline fairness regularizer. We consider quadrilateral patterns $\mathbf{p}_{i,j}$, $i, j \in \mathbb{Z}$ on the (x, y)-plane, which are liftable to a paraboloid S, while maintaining face planarity. We define S as $z = \kappa_1 x^2 + \kappa_2 y^2$, $\kappa_1 \kappa_2 \neq 0$. Following Appendix A, we consider the induced quadratic form $\langle \mathbf{a}, \mathbf{b} \rangle := \kappa_1 x_a x_b + \kappa_2 y_a y_b$ on the (x, y)-plane, the squared norm $|\mathbf{a}|^2 = \langle \mathbf{a}, \mathbf{a} \rangle$, and the conjugacy relation $\langle \mathbf{a}, \mathbf{b} \rangle = 0$.

Polyline fairness Straight polylines hold:

$$\begin{aligned} \frac{1}{2}(\mathbf{p}_{i+1,j} + \mathbf{p}_{i-1,j}) &= \mathbf{p}_{i,j}, \\ \frac{1}{2}(\mathbf{p}_{i,j+1} + \mathbf{p}_{i,j-1}) &= \mathbf{p}_{i,j}. \end{aligned} \tag{B.1}$$

These conditions are essentially symmetries with respect to vertices. An equivalent formulation is:

$$\begin{aligned} \mathbf{p}_{i+1,j} - \mathbf{p}_{i,j} &\equiv \mathbf{u}, \\ \mathbf{p}_{i,j+1} - \mathbf{p}_{i,j} &\equiv \mathbf{v}, \end{aligned} \tag{B.2}$$

where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ are the respective edge vectors. With perfect straight polylines of equally distributed vertices, the quadrilaterals of the tiling are congruent parallelograms which are translations of each other. When $\mathbf{u}$, $\mathbf{v}$ and a single vertex is given, the whole pattern is fixed.

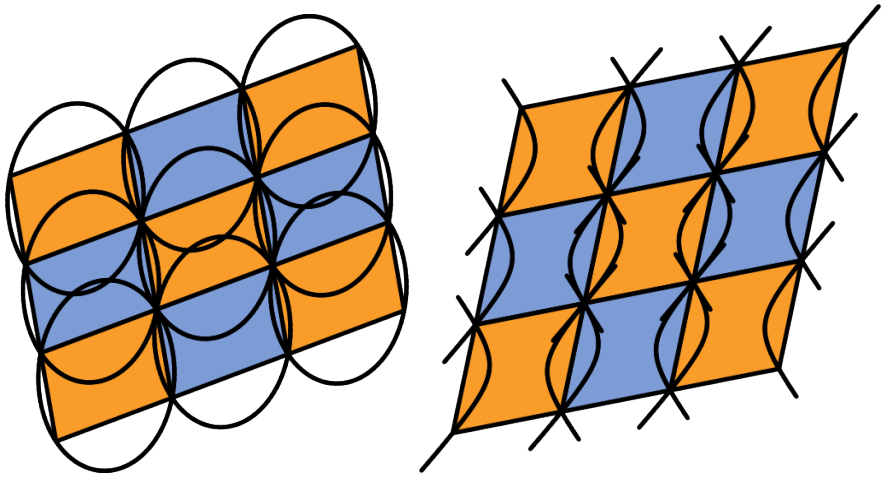


Figure B.1: Polyline fairness requires a quad mesh to be initialized according to conjugate directions.

In order to be liftable to a paraboloid while maintaining planarity, every parallelogram must be inscribed to a conic in the form of $|\mathbf{p} - \mathbf{c}|^2 = \gamma,\ \gamma \in \mathbb{R}$, as shown in the Appendix A. This requires $\mathbf{u}$ and $\mathbf{v}$ to be conjugate: $\langle \mathbf{u}, \mathbf{v} \rangle = 0$.

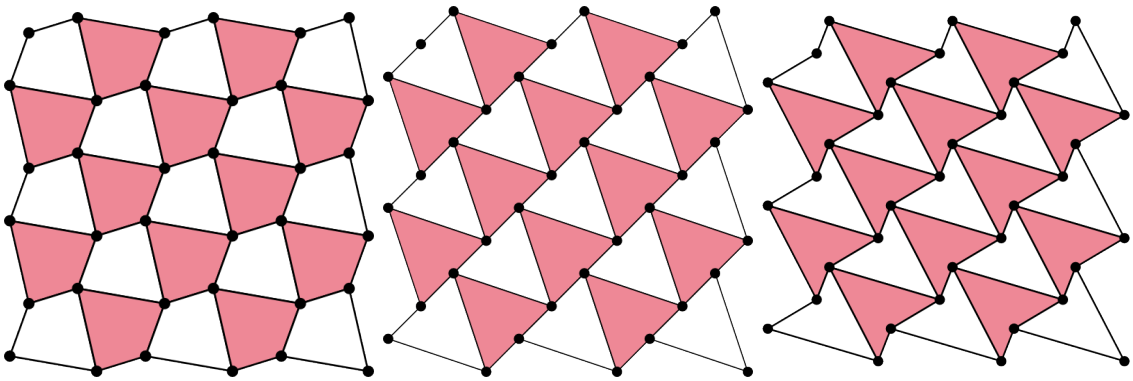


Figure B.2: Imposing edge-midpoint symmetries introduces two more degrees of freedom in comparison with polyline fairness.

For the hyperbolic case, $\kappa_1\kappa_2 < 0$, the asymptotic directions $\mathbf{d} := \lambda(\sqrt{\kappa_2}, \pm\sqrt{\kappa_1})$ are self-conjugate, as $\langle \mathbf{d}, \mathbf{d} \rangle = \kappa_1 x_{\mathbf{d}} x_{\mathbf{d}} + \kappa_2 y_{\mathbf{d}} y_{\mathbf{d}} = \kappa_1|\kappa_2| + \kappa_2|\kappa_1| = 0$. When one family of polylines is aligned in a direction close to an asymptotic direction, the other family of polylines must also be aligned with a close direction, which results in ill-shaped narrow quads. From this we draw that the polyline fairness is only effective for conjugate directions which are close to orthogonal, i.e., principal directions, and are far from asymptotes (See Figure B.1).

Edge midpoint symmetry Symmetries with respect to edge midpoints require each pair of adjacent faces to be symmetric with respect to the midpoint of their common edge:

$$\begin{aligned} \frac{1}{2}(\mathbf{p}_{i\pm1,j} + \mathbf{p}_{i\mp1,j+1}) &= \frac{1}{2}(\mathbf{p}_{i,j} + \mathbf{p}_{i,j+1}), \\ \frac{1}{2}(\mathbf{p}_{i,j\pm1} + \mathbf{p}_{i+1,j\mp1}) &= \frac{1}{2}(\mathbf{p}_{i,j} + \mathbf{p}_{i+1,j}). \end{aligned} \tag{B.3}$$

By linear combinations, those conditions are equivalent to:

$$\begin{aligned} \mathbf{p}_{i+1,j+1} - \mathbf{p}_{i,j} &\equiv \mathbf{s}, \\ \mathbf{p}_{i+1,j-1} - \mathbf{p}_{i,j} &\equiv \mathbf{t}, \end{aligned} \tag{B.4}$$

where $\mathbf{s}, \mathbf{t} \in \mathbb{R}^2$ are the diagonal vectors of every quad in the mesh.

This produces two more degrees of freedom for quad tilings, in comparison with polyline fairness, since the actual positions of the diagonals can vary, and the quad is not constrained to be a parallelogram anymore (See Figure B.2). Effectively, the quadrilateral pattern is naturally decomposed to two independent sets of vertices. These degrees of freedom are apparent on computed results with general surfaces as well.

Appendix C

Quadratic Projection

In the procedure of *Guided Projection*, the quadratic equations need to be approximated by linear equations at the beginning of each iteration. In the discussion above, the linearization step is based on Taylor expansions, following the original Newton's methods. Geometrically, a surface could also be approimated by a tangent plane locally. Therefore, the following discussion proposes an alternative way to linearize quadratic equations based on projecting to quadratic hypersurfaces and taking tangent hyperplanes at the footpoints.

Implementing closest point projection onto a quadratic surface. We discuss how to efficiently evaluate the closest point projection $\mathbf{q}$ of a point $\mathbf{p}$ onto a quadratic surface Φ given by the implicit equation $\mathbf{x}^\top H\mathbf{x} = \alpha$, with a symmetric matrix H.

$$\mathbf{q} \text{ minimizes } \|\mathbf{p} - \mathbf{x}\|^2, \text{ such that } \mathbf{x} \in \Phi.$$

This constrained optimization problem is not difficult. A standard compactness argument shows existence of a closest point $\mathbf{p}$, and smoothness of the functions involved shows it can be found by the Lagrange multiplier method. We use the following procedure:

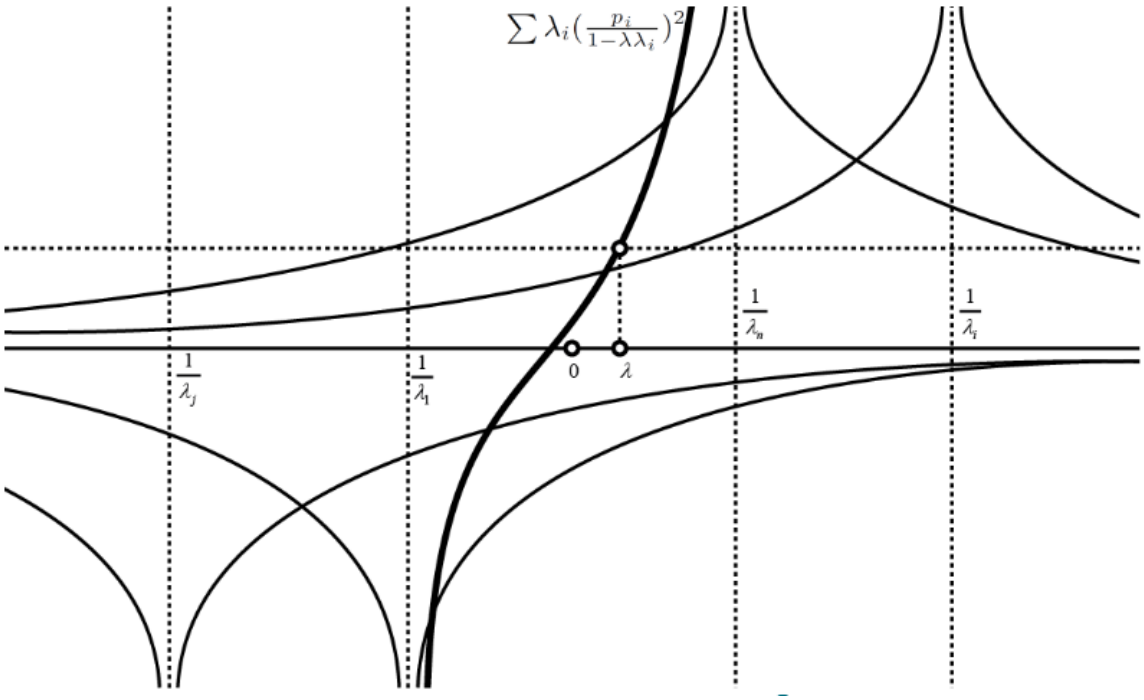


Figure C.1: Illustration of the location of the Lagrangian multipliers and the identification of the multiplier corresponding to the closest foot-point.

- We change to a coordinate system defined by an orthonormal basis of eigenvectors of H (in our applications this is fast, since the number of occupied places in H is bounded via the degree of vertices in meshes). Φ is then defined by $\mathbf{x}^\top H\mathbf{x} = \alpha$, with

$$H = \mathrm{diag}(\lambda_1, \ldots, \lambda_n), \quad \lambda_1 \leq \ldots \leq \lambda_n.$$

By inverting some coordinate axes we achieve that the coordinates $(p_1, \ldots, p_n)$ of $\mathbf{p}$ are nonnegative.

- Assume we have found a closest point $\mathbf{q} = (q_1, \ldots, q_n)$. If $q_i < 0$, then $\mathbf{q}' = (q_1, \ldots, -q_i, \ldots, q_n)$ is contained in Φ but lies closer to $\mathbf{p}$ than $\mathbf{q}$ does:

$$\|\mathbf{p} - \mathbf{q}\|^2 - \|\mathbf{p} - \mathbf{q}'\|^2 = (p_i - q_i)^2 - (p_i + q_i)^2 = -4p_iq_i.$$

If $p_i > 0$ this is positive, contradicting that $\mathbf{q}$ is closest (if $p_i = 0$ then a closest point

has $q_i \geq 0$ by symmetry). Thus, $q_i \geq 0$.

- The location of $\mathbf{q}$ is found by introducing a Lagrangian multiplier λ and comparing gradients of the constraint $\mathbf{x}^\top H\mathbf{x}$ and the target function $\|\mathbf{p}-\mathbf{x}\|^2$: $\mathbf{q}$ must fulfill $\lambda H\mathbf{q} = \mathbf{q}-\mathbf{p}$:

$$\lambda\lambda_i q_i = q_i - p_i \implies q_i = \frac{p_i}{1-\lambda\lambda_i}.$$

Since $p_i, q_i \geq 0$ we get $1 \geq \lambda\lambda_i$. This inequality is to hold for all eigenvalues λ_i. Depending on the signs of eigenvalues λ_i, these inequalities are summarized as follows:

$$\begin{aligned}
&\lambda \in [\tfrac{1}{\lambda_1}, \infty) && \text{if} \quad \lambda_n \leq 0,\\
&\lambda \in [\tfrac{1}{\lambda_1}, \tfrac{1}{\lambda_n}] && \text{if} \quad \lambda_1 < 0 < \lambda_n,\\
&\lambda \in (-\infty, \tfrac{1}{\lambda_n}] && \text{if} \quad 0 \leq \lambda_1.
\end{aligned}$$

- Lastly we determine $\mathbf{q}$ by the condition $\mathbf{q}^\top H\mathbf{q} = \alpha$, which means solving the equation $\sum \lambda_i (\frac{p_i}{1-\lambda\lambda_i})^2 = \alpha$ for λ. Differentiation shows that the l.h.s. of this equation is a strictly increasing function of λ (indeed, a sum of strictly increasing summands), provided λ is contained in the interval specified above; further the l.h.s. is unbounded as λ approaches either end of the interval. It follows that there is a unique solution λ, as illustrated in Figure C.1, and thus a unique closest point $\mathbf{q}$.