Die approbierte Originalversion dieser Dissertation ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

Dissertation

Special Curve Patterns for Freeform Architecture

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

O.Univ.-Prof. Dr. Helmut Pottmann
Institut für Diskrete Mathematik und Geometrie (E104)

eingereicht an der Technischen Universität Wien,
Fakultät für Mathematik und Geoinformation,
von

Bailin Deng, MSc.
Matr.Nr.: 0829009
Tigergasse 23-27/3/2, 1080 Wien

Wien, am 29.08.2011

Zusammenfassung

In den letzten Jahren wurden Freiform-Flächen in der Architektur immer beliebter. Die technischen Herausforderungen, die bei der Realisierung solcher Flächen zu bewältigen sind haben ein neues, aktives Forschungsfeld – die Architekturgeometrie – geschaffen. In dieser Arbeit untersuchen wir spezielle Kurvenmuster auf Flächen, die im Design und der Realisierung architektonischer Freiform-Flächen Anwendung finden.

Zuerst betrachten wir Familien (stückweise) geodätischer Kurven auf Flächen die zur Einteilung in Paneele und beim Innenraum Entwurf wichtig sind. Wir schlagen ein Verfahren zur Ausbreitung solcher Kurven ausgehend von einer Startkurve vor, so dass die Distanzfunktion benachbarter Kurven eine gegebene Distanzfunktion annährt. Als lineare Approximation von Distanzfunktionen verwenden wir Jacobi-Felder. Eine Nachbarkurve wird dann entsprechend des ausgewählten Jacobi-Feldes durch Lösen eines Optimierungsproblems bestimmt. Durch Verwendung verschiedener Distanzfunktionen können unterschiedliche Kurvenmuster erzeugt werden. Die vorgestellte Methode erlaubt den intuitiven und leicht handhabbaren Entwurf von goedätischen Mustern auf Freiform-Flächen.

Im Anschluss wird eine Methode zur Berechnung funktionaler Gewebe vorgestellt. Ein funktionales Gewebe besteht aus drei Kurvenfamilien mit regulärer Konnektivität, wobei die Kurven zusätzliche funktionale Eigenschaften erfüllen müssen. Um den Herstellungsprozess der einzelnen Kurvenelemente zu verbessern, werden ebene, zirkuläre und geodätische Kurvensegmente betrachtet. Die diskrete Entsprechung eines Gewebes ist ein reguläres Dreiecksnetz. Dabei entsprechen die Kurven des Gewebes Kantenzügen des Netzes. Die Gestalt des Gewebes wird durch ein Funktional bestimmt, das die Abweichung der Kurven von geforderten Eigenschaften bestraft. Des Weiteren wird gezeigt, dass Gewebe die aus ebenen Kurven bestehen unter Verwendung dreier Familien von Ebenen exakt berechnet werden können. Durch die Bereitstellung eines Verfahrens zum Entwurf von Geweben bestehend aus leicht herzustellenden Kurvensegmenten, liefert diese Arbeit einen Beitrag zur Realisierung von Geweben wie sie in aktuellen architektonischen Entwürfen zu finden sind.

Abstract

In recent years, freeform shapes are gaining more and more popularity in architecture. Such shapes are often challenging to manufacture, and have motivated an active research field called architectural geometry. In this thesis, we investigate patterns of special curves on surfaces, which find applications in design and realization of freeform architectural shapes.

We first consider families of geodesic curves or piecewise geodesic curves on a surface, which are important for panelization of the surface and for interior design. We propose a method to propagate a series of such curves across a surface, starting from a given source curve, so that the distance functions between neighboring curves are close to given target distance functions. We use Jacobi fields as first order approximation of the distance functions from a curve to its neighboring curves, and select a Jacobi field which is closest to the target distance function. A neighboring curve is then computed according to the selected Jacobi field by solving an optimization problem. Using different target distance functions, we can generate different patterns of geodesic/piecewise geodesic curves. Our method provides an intuitive and controllable way to design geodesic patterns on freeform surfaces.

We then present a method to compute functional webs, which are three families of curves with regular connectivity, where the curves have given special properties. We consider planar, circular and geodesic properties of the curves, which facilitate the fabrication of curve elements. We discretize a web as a regular triangle mesh, where the curves are represented by edge polylines of the mesh. The shape of the web is determined by optimizing a target functional which penalizes the deviation of the curves from their target properties. Furthermore, for webs where all curves are planar, we also show they can be computed in an exact way using three families of planes. By enabling the design of webs composed of curve elements which are easily manufacturable, our method addresses the challenge in realization of webs which have emerged in recent architectural designs.

Acknowledgments

I would like to thank all the people who have supported me during my doctoral study.

I am grateful to my supervisor Helmut Pottmann, who offered me the opportunity to work on architectural geometry, and has always provided guidance and support. His profound knowledge of geometry, and his elegant way of applying it to real-world applications, have set an example for me. I would like to express my appreciation to Johannes Wallner for his advice on my work, and for his exceptional writing skill. I would also like to thank Niloy Mitra for taking the job as external reviewer.

This thesis would not have been completed without the help from my colleagues. I am thankful to Christian Müller and Martin Kilian for their advice on improving the thesis, to Heinz Schmiedhofer and Martin Reis for rendering Figures 3.12, 3.14, 3.18, and 3.20, and to Martin Kilian for translating the abstract into German.

Throughout my study, I learned a lot from the discussions with the colleagues in our institute, including, but not limited to, Bernhard Blaschitz, Simon Flöry, Philipp Grohs, Mathias Höbinger, Martin Kilian, Friedrich Manhart, Christian Müller, Boris Odehnal, Martin Peternell, and Alexander Schiftner. I am grateful to them for sharing their knowledge and expertise.

My work has been supported by the Austrian Science Fund (FWF) under the grant S9206.

Finally I would like to thank my parents for their unconditional love and support. Many thanks to Xianbin, for being the greatest friend; and to Spencer, for her understanding and love.

Contents

1 Introduction

In recent years, freeform shapes are becoming more and more popular in architecture. Following the Frank Gehry's pioneering work in adapting digital technologies to architectural design [Lindsey 2001] [Mitchell 2001], architects are now able to employ computer-aided design (CAD) tools in the design process, which allow them to specify complicated shapes in an accurate and easy way. The use of digital design tools, together with the technological advances in material and construction, have enabled the use of freeform elements in contemporary architecture (see Figures 1.1 and 1.2 for some examples of freeform architecture). In order to realize a complex freeform shape in a feasible and affordable way, the designed surface is usually segmented into smaller components which are easier to fabricate. For example, the freeform exterior surface of the Walt Disney Concert Hall in Figure 1.1 is realized using a large number of stainless steel panels (see Figure 1.3). This process of approximating a designed surface using simple elements is called *rationalization*. For complicated freeform surfaces, rationalization is a challenging task. First of all, the simple elements used in rationalization usually lead to deviation between the designed surface and the approximation surface. To guarantee

Figure 1.1: Walt Disney Concert Hall, Los Angeles, designed by Frank Gehry. Photo by Carol Highsmith.

Figure 1.2: Hungerburg Station, Innsbruck, Austria, designed by Zaha Hadid. Photo by Francis Wu via Flickr.

Figure 1.3: Details of the freeform exterior surface of the Walt Disney Concert Hall, which is realized using stainless steel panels. Photos by Renee Silverman and Richard Banks via Flickr.

Figure 1.4: Centre Pompidou-Metz, France, designed by Shigeru Ban. The roof contains an interwoven curve structure made from glued laminated timber. Photos by Didier Boy de la Tour.

consistency between the realized shape and the original design, this deviation needs to be small. Furthermore, for nice visual appearance of the realized shape, the elements of the approximation surface need to be arranged in an aesthetically pleasing way, which further limit the available degrees of freedom for the approximation surface. In general, rationalization cannot be handled by standard CAD tools. This limitation has motivated research efforts from architects, engineers, mathematicians, and computer scientists, which have now become an active research field called *architectural geometry*. Introduction to this field and recent progress can be found in the textbook [Pottmann et al. 2007a], articles [Pottmann et al. 2008c] [Pottmann 2009] [Pottmann 2010] [Wallner and Pottmann 2011], and conference proceedings [Pottmann et al. 2008a] [Ceccato et al. 2010]. Architectural geometry aims at geometric modeling and optimization tools for architectural needs, especially for economical realization of freeform architectural designs. Previously, most research work focused on rationalization methods with the approximation surface composed of various simple elements such as planar quadrilaterals [Glymph et al. 2004] [Liu et al. 2006] [Zadravec et al. 2010], planar hexagons [Wang et al. 2008], single curved panels [Pottmann et al. 2008b], circles [Schiftner et al. 2009], ruled surface patches [Flöry and Pottmann 2010], repeated quads or triangles [Fu et al. 2010] [Singh and Schaefer 2010], and panels produced from reused molds [Eigensatz et al. 2010]. Other work is concerned with the optimization of multi-layer structures [Pottmann et al. 2007b] and circular arc structures [Bo et al. 2011], which lead to easily manufacturable beams and nodes.

In this thesis, we are instead interested in curve families on a surface. The importance of curve families in freeform architecture is two-fold. Firstly, starting from a family of curves on a surface, we can compute a set of single curved panels covering the surface using the optimization algorithm from [Pottmann et al. 2008b], where the curves are taken as the initial boundaries between the panels. Since this is a highly nonlinear problem, the initial curves must be arranged properly to guarantee the success of the optimization approach. For example, to compute a geodesic strip model, an initial set

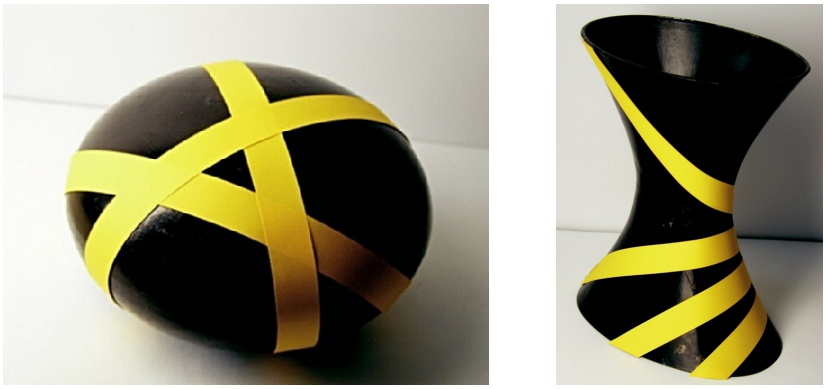


Figure 1.5: If a straight paper strip is bent to lie on a surface, it will follow a geodesic on the surface.

of geodesic curves is needed [Pottmann et al. 2008b]. Therefore, curve families play an important role in the panelization of a freeform surface. Secondly, curve families can be part of a freeform architectural design (see Figure 1.4 for an example). The layout of these curves are essential for the visual appearance as well as the structural integrity of the design.

In architecture, some special curves have nice properties in terms of structural functionality and manufacturing cost. One example is planar curves in vertical planes, which can be used as support elements. Another example is geodesic curves. A geodesic curve is a generalization of the notion of a straight line to surfaces [do Carmo 1976]. If we take a straight strip made from paper or wood, and bend it to follow the shape of a surface, the bent strip will follow a geodesic on the surface (see Figure 1.5). A family of geodesic curves on a surface can represent a set of developable strips covering the surface, with each strips having approximately straight development [Pottmann et al. 2008b]. These strips can be easily fabricated by trimming standard-size rectangular strips. If a curve element is a geodesic, it can be manufactured by bending a straight timber board about its weak axis, which is inexpensive compared with other manufacturing approaches like CNC machining [Pirazzi and Weinand 2006]. In this thesis, we investigate families of special curves (such as geodesics) on freeform surfaces, and propose computational tools to create such families. The investigation is organized as follows:

- In Chapter 2, we are concerned with geodesic curves and piecewise geodesic curves with approximately constant distance between each other, which can represent wooden panels covering a surface, with each panel having approximately straight development and small variation of width. Such panel layouts are useful for realization of freeform architectural shapes, as well as for interior design. We first review the Jacobi equation, which governs the distance between two neighboring geodesics. Using the Jacobi equation, we propose methods to evolve a family of geodesic/piecewise geodesic curves starting from a given source curve, so that the distance functions between neighboring curves are close to given target distance

functions.

- In Chapter 3, we are interested in curve structures consisting of three families of curves with regular connectivity (like Figure 1.4), which we call *webs*. Such structures have appeared in some recent architectural designs, and might be expensive to build due to the intricate shapes of the curve elements. We first show the connection between these structures and a classical geometry research topic called *web geometry*. Then we provide algorithms to compute webs consisting of special curve elements, including planar, circular and geodesic curves, which can either be manufactured in an inexpensive way, or have desirable structural properties. We do so by solving a non-linear optimization problem, using a target functional which takes the target properties of the curves into account. For webs consisting of only planar curves, we show how they can be computed in an exact way using three families of planes.

In both chapters, the curve families can be regarded as patterns on a surface. In other words, we are investigating patterns of special curves on freeform surfaces.

2 Geodesic Patterns on Freeform Surfaces

In this chapter, we consider the problem of cladding a general double curved surface with straight panels. More precisely, given a double curved surface, we look for a set of developable strips with nearly straight development, so that the strips can be bent and joined to approximate the given surface. This problem has applications in freeform architectural design, especially for interior spaces (see Figures 2.1, 2.2, 2.3). It can be shown that such panels follow geodesic curves on the given surface [Pottmann et al. 2008b]. In practice, these panels are often trimmed from rectangular strips of fixed width d. To reduce material waste, we want to have panels whose widths are close to d. Such a problem can be solved by searching for a family of geodesic curves on the surface with the intrinsic distance between neighboring geodesic curves close to d. In this chapter, we solve a more general problem:

Problem 2.1. *Given a freeform surface and a target distance function $W(s)$, compute a family of geodesics $\{g_i\}$ on the surface such that the distance function $w_i(s)$ from g_i to g_{i+1} is close to $W(s)$, where s is the arc-length parameter of g_i.*

With this formulation, we can not only generate geodesic curves with almost constant distance between neighboring curves, but also create interesting patterns of geodesics by using special distance functions $W(s)$.

Problem 2.1 is not easy to solve. In general, the intrinsic distance between two points on a surface can not be expressed by a formula, except for some special surfaces such as surfaces of constant Gaussian curvature and surfaces of revolution [do Carmo 1976]. So in practice, surface cladding with straight panels can be done only for surfaces of special shapes (see Figures 2.1, 2.2), or in an experimental way (see Figure 2.3). In this chapter we employ a first order approximation of the distance between two neighboring geodesics, which enables us to solve Problem 2.1 for general freeform surfaces.

We assume that the freeform surface is given as a triangular mesh. And the resulting geodesics will be represented as polylines on the mesh surface. The surface needs to be of disk topology. We will not talk about how to compute actual panels from the geodesic curves. Such a problem is addressed by [Pottmann et al. 2008b] and [Wallner et al. 2010].

2.1 Related Work

A geodesic curve is a locally shortest path on a surface S. The computation of geodesics is a classical topic. Depending on the given conditions, there are basically two types of problems. The first one is an *initial value problem*: given a point $p \in S$ and a vector $v \in T_pS$ where T_pS is the tangent space of S at p, find a geodesic g which is incident with p, such that the tangent vector of g at p is v. The second one is a *boundary*

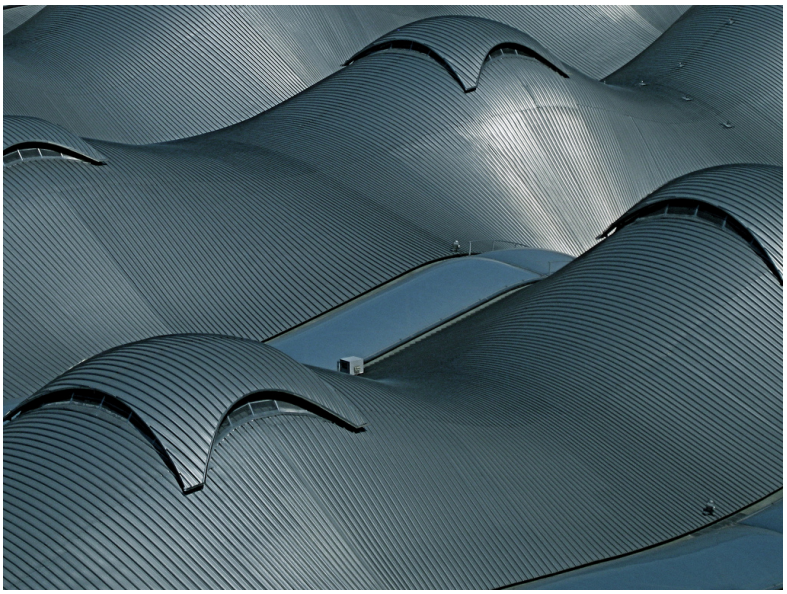


Figure 2.1: The roof of Southern Cross Station at Melbourne, Australia, covered with straight metal panels. Photo by Dale Gillard via Flickr.

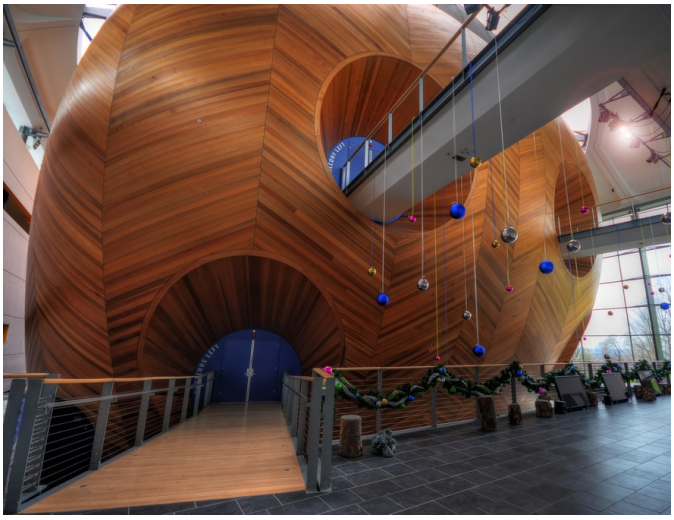
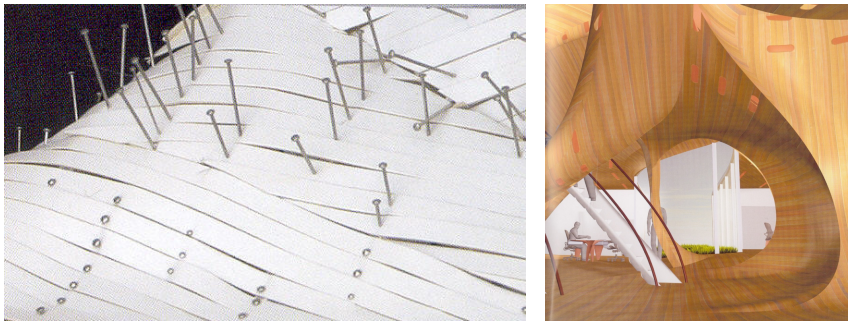


Figure 2.2: Interior design of the experimental media and performing arts center at Rensselaer Polytechnic Institute, Troy, New York, using straight wooden panels. Photo by Skunkworks Photographic via Flickr.



(a) (b)

Figure 2.3: Experimental cladding using paper strips (a) results in an office space design by NOX Architects ((b), see [Spuybroek 2004]).

value problem: given two points $p_1, p_2 \in S$, find a geodesic g connecting p_1 and p_2. For smooth surfaces, the first problem is equivalent to solving an initial value problem for a second order ODE, while the second problem can be converted into the constrained minimization of the quadratic energy $\int \|\dot{g}\|^2$ [do Carmo 1976].

On polyhedral surfaces, geodesic curves are polylines. [Polthier and Schmies 1998] propose *straightest geodesics* as solutions to the initial value problem on polyhedral surfaces. If a straightest geodesic g crosses a surface edge E, the two segments of g which are incident with E will form equal angles with E; if g crosses a surface vertex V, the two segments incident with V will bisect the sum of angles from surface faces incident with V. The time complexity of computing a straightest geodesic is $O(n)$, where n is the number of edges of the surface. The boundary value problem of computing geodesics on polyhedral surfaces is closely related to the computation of shortest paths between two points on a surface. Since shortest path problems find applications in a wide range of areas such as computer graphics and robotics, many algorithms have been proposed to compute exact or approximate shortest paths between two points. [Mitchell et al. 1987] provide an algorithm (MMP) to compute the exact shortest path from a single source point to any point on a polyhedral surface. They observe that each edge of the surface can be divided into a set of intervals (also called *windows*), over which the shortest distance function to the source point can be computed atomically. The MMP algorithm computes these windows by propagating the distance information over the surface, using a Dijkstra-like method. It is of time complexity $O(n^2 \log n)$ and space complexity $O(n^2)$. Using these windows, a shortest path from the source point to any point on the surface can be traced with time complexity $O(k + \log n)$ where k is the number of faces crossed by the path. [Surazhsky et al. 2005] show that despite the worst case complexity $O(n^2 \log n)$, the MMP algorithm can run much faster in practice. They propose an efficient implementation of the MMP algorithm. [Liu et al. 2007] discuss degenerate cases that might appear in the algorithm of [Surazhsky et al. 2005], and provide algorithms to handle such cases. [Chen and Han 1990] propose an algorithm

(CH) to construct a *sequence tree* to store the information about the distance from the source point to each edge. Unlike the MMP algorithm, they do not employ Dijkstra-like propagations. The CH algorithm is of time complexity $O(n^2)$ and space complexity $\Theta(n)$. [Kaneva and O'Rourke 2000] propose an implementation of the CH algorithm. [Xin and Wang 2009] improve the CH algorithm by filtering useless windows, and by maintaining a priority queue like Dijkstra's algorithm. Their implementation significantly outperforms the original CH algorithm.

Exact geodesic curves between two points are usually expensive to compute. Therefore, approximate geodesic curves are often used in practice. [Kimmel and Sethian 1998] compute approximate geodesic paths on triangular meshes by solving the eikonal equation with the fast marching method (FMM). The running time of their algorithm is $O(n \log n)$. Various improvements for this method have been proposed ([Novotni and Klein 2002][Kirsanov 2004][Yatziv et al. 2006]) to achieve lower time complexity or higher accuracy. [Martínez et al. 2005] and [Xin and Wang 2010] propose iterative methods to improve the geodesic paths computed by FMM.

For the problem of cladding freeform surfaces with developable panels, early research evolved from the architecture of F.Gehry [Shelden 2002]. [Pottmann et al. 2008b] investigate the problem of covering freeform surfaces by developable strips. They introduce *geodesic strip models*, which are continuous surfaces composed of developable strips with nearly straight development. This is closely related to the problem we want to solve here. In their paper the geodesic strip models are computed by numerical optimization requiring careful initialization, and there is no direct control of the strip width. Instead, the method in this chapter aims at producing geodesic curves with distance functions between neighboring curves close to prescribed functions. Such a formulation provides more flexibility in controlling the shapes of the strips. Besides, the geodesics computed by our methods can be used to initialize the optimization algorithm in [Pottmann et al. 2008b]. Recently, [Kahlert et al. 2011] study the tiling of a surface by strips of bounded width whose boundary curves are quasi-geodesics [Aleksandrov and Zahlgaller 1967], which is similar to the problem we solve in this chapter. They only aim at nearly-constant strip widths. Besides, for surface regions with high Gaussian curvature it may not be possible to find two neighboring geodesics without intersection. For such surfaces [Kahlert et al. 2011] can only generate quasi-geodesics with intersections, which might not be desirable in real applications. The *piecewise geodesic curves* introduced in this chapter can handle such cases and generate strip boundary curves without intersection.

2.2 Distance Between Geodesics

Although closed-form representations of the intrinsic distance between two points are in general not available, there is a well-known first order approximation of the intrinsic distance between two neighboring geodesics: Start at time $t = 0$ with a geodesic curve $g(s)$, which is parametrized by arc length s, and let it move smoothly with time. A snapshot at time $t = \varepsilon$ yields a geodesic curve g^+ near g (see Figure 2.4):

$$g^+(s) = g(s) + \varepsilon \mathbf{v}(s) + \varepsilon^2(\ldots).$$

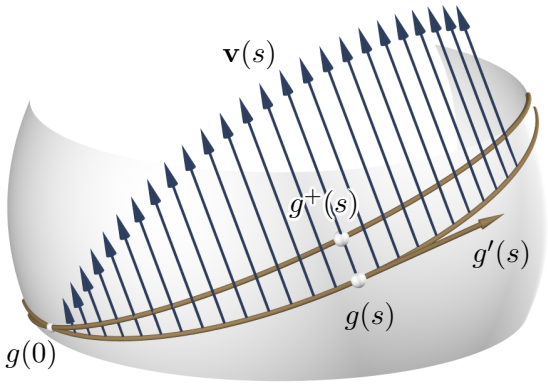


Figure 2.4: A geodesic $g(s)$ with Jacobi field $\mathbf{v}(s)$, and a neighboring geodesic $g^+(s)$ which is at distance $\approx \varepsilon\|\mathbf{v}(s)\|$. The parameter s is the arc length along the geodesic g, and $\mathbf{v}$ obeys the Jacobi differential equation (2.1). Here $\mathbf{v}(0) = \mathbf{0}$.

The derivative vector field $\mathbf{v}(s)$ is called a Jacobi field. It is known that without loss of generality we may assume $\mathbf{v}(s)$ is orthogonal to the curve $g(s)$, and it can be expressed in terms of the geodesic's tangent vector g' as

$$\mathbf{v}(s) = w(s) \cdot R_{\pi/2}(g'(s)), \tag{2.1}$$

where

$$w''(s) + K(s)w(s) = 0. \tag{2.2}$$

Here R_α is the rotation by angle α in the tangent plane of the surface, and $K(s)$ is the Gaussian curvature function along $g(s)$ [do Carmo 1976].

To make use of the Jacobi fields, we define the signed distance function from g to g^+ as follows (see Figure 2.5).

Definition 2.1. *Let g and g^+ be two neighboring geodesic curves. For a point $p \in g$, trace a geodesic from p with initial tangent direction $R_{\pi/2}(T(p))$, where $T(p)$ is the unit tangent vector of g at p. Let $h(t)$ be an arc-length parametrization of this geodesic, such that the tangent vector of $h(t)$ at p is $R_{\pi/2}(T(p))$. Let X be the intersection point between h and g^+. Then the signed distance from p to g^+ is defined as*

$$\mathcal{D}(p, g^+) = t_1 - t_0,$$

where $h(t_0) = p$, $h(t_1) = X$. And the signed distance function from $g(s)$ to g^+ is defined as

$$\mathcal{W}_g(s) = \mathcal{D}(g(s), g^+).$$

Now the signed distance function $\mathcal{W}_g(s)$ in Definition 2.1 can be approximated by a function $w(s)$ satisfying Equation (2.2). And Problem 2.1 is converted to a problem of finding such a function $w(s)$ which is close to the target distance function $W(s)$.

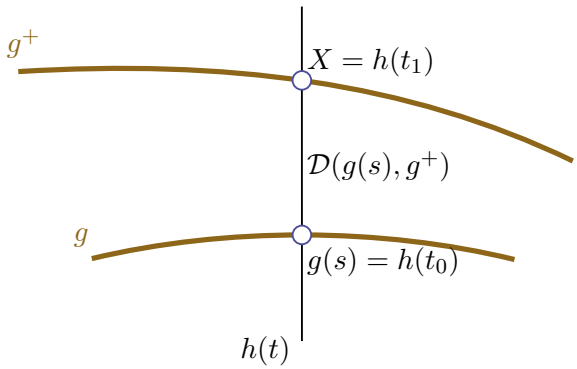


Figure 2.5: The signed distance $\mathcal{D}(g(s), g^+)$ from $g(s)$ to g^+ is computed with a geodesic $h(t)$ which is orthogonal to g.

2.3 Evolving Geodesics on Mesh Surfaces

2.3.1 Solving the Jacobi Equation

Since the Jacobi equation (2.2) is a second order linear ODE with coefficient function $K(s)$, the solutions form a two dimensional linear space. Any solution $w(s)$ is a linear combination of two linearly independent fundamental solutions $w^{(1)}(s), w^{(2)}(s)$

$$w(s) = \lambda_1 w^{(1)}(s) + \lambda_2 w^{(2)}(s). \tag{2.3}$$

The fundamental solutions are found as follows: The geodesic g under consideration is sampled at arc-length parameter values $x_0 < x_1 < \ldots < x_M$. The unknown function $w(s)$ is represented by its values $w_i = w(x_i)$, $i = 0, \ldots, M$ at the sample points. The second derivative w'' at x_i is approximated with finite difference

$$w_i'' = \frac{2}{L_i + L_{i+1}}\left(\frac{w_{i+1} - w_i}{L_i} - \frac{w_i - w_{i-1}}{L_{i-1}}\right), \quad i = 1, \ldots, M-1,$$

where $L_i = x_{i+1} - x_i$. Then at sample parameter x_i, Equation (2.2) becomes

$$\frac{2}{L_i + L_{i+1}}\left(\frac{w_{i+1} - w_i}{L_i} - \frac{w_i - w_{i-1}}{L_{i-1}}\right) + K_i w_i = 0, \quad i = 1, \ldots, M-1, \tag{2.4}$$

where $K_i = K(x_i)$ is the Gaussian curvature at point $g(x_i)$. This is a linear system. And the solution can be written as a recurrence relation

$$w_{i+1} = \frac{(2 - L_{i-1}L_iK_i)(L_{i-1} + L_i)}{2L_{i-1}} w_i - \frac{L_i}{L_{i-1}} w_{i-1}, \quad i = 1, \ldots, M-1. \tag{2.5}$$

Then for given initial values w_0, w_1, we can compute $w_i (i = 2, \ldots, M)$ by Equation (2.5). For fundamental solutions $w^{(1)}$ and $w^{(2)}$, we simply use initial values $w_0^{(1)} = 0, w_1^{(1)} = 1$ and $w_0^{(2)} = 1, w_1^{(2)} = 0$.

Another way to solve for the fundamental solutions is to impose boundary conditions $w_0^{(1)} = 0, w_M^{(1)} = 1$ and $w_0^{(2)} = 1, w_M^{(2)} = 0$ to linear system (2.4). The unknown values $w_1^{(i)}, \ldots, w_{M-1}^{(i)}$ $(i = 1, 2)$ can be computed by solving a tridiagonal linear system

$$\mathbf{MW} = \mathbf{B}, \tag{2.6}$$

where

$$\begin{aligned}
\mathbf{M} &= \begin{bmatrix}
L_0L_1K_1-2 & \frac{2L_0}{L_0+L_1} & & & \\
\frac{2L_2}{L_1+L_2} & L_1L_2K_2-2 & \frac{2L_1}{L_1+L_2} & & \\
 & \ddots & \ddots & \ddots & \\
 & & \frac{2L_{M-2}}{L_{M-3}+L_{M-2}} & L_{M-3}L_{M-2}K_{M-2}-2 & \frac{2L_{M-3}}{L_{M-3}+L_{M-2}} \\
 & & & \frac{2L_{M-1}}{h_{M-2}+h_{M-1}} & h_{M-2}h_{M-1}K_{M-1}-2
\end{bmatrix}, \\
\mathbf{W} &= [w_1^{(i)}, w_2^{(i)}, \ldots, w_{M-2}^{(i)}, w_{M-1}^{(i)}]^T, \\
\mathbf{B} &= [-\frac{2L_1w_0^{(i)}}{L_0+L_1}, 0, \ldots, 0, -\frac{2L_{M-2}w_M^{(i)}}{L_{M-2}+L_{M-1}}]^T.
\end{aligned}$$

This type of fundamental solution will be used in Section 2.4. Note that linear system (2.6) may be singular and we may not get $w^{(1)}$, $w^{(2)}$ from the given boundary conditions. This can happen when $g(s_M)$ is conjugate to $g(s_0)$ relative to g, which means that there exists a non-zero solution $w^{(c)}(s)$ of the Jacobi equation with $w^{(c)}(s_0) = w^{(c)}(s_M) = 0$ [do Carmo 1976].

2.3.2 Selecting a Jacobi Field

To select a Jacobi field closest to a target distance function $W(s)$ on an interval I, we compute the coefficients λ_1, λ_2 in Equation (2.3) by minimizing the error function

$$F_w = \int_I [w(s) - W(s)]^2 ds = \int_I [\lambda_1 w^{(1)}(s) + \lambda_2 w^{(2)}(s) - W(s)]^2 ds.$$

This problem reduces to solving a 2×2 linear system. The solution is

$$\begin{aligned}
\lambda_1 &= \frac{D_1E_{22} - D_2E_{12}}{E_{11}E_{22} - E_{12}^2}, \\
\lambda_2 &= \frac{D_2E_{11} - D_1E_{12}}{E_{11}E_{22} - E_{12}^2},
\end{aligned}$$

where

$$\begin{aligned}
E_{ij} &= \int_I w^{(i)}(s)\ w^{(j)}(s)\ ds, \quad i, j = 1, 2, \\
D_i &= \int_I w^{(i)}(s)\ W(s)\ ds, \quad i = 1, 2.
\end{aligned}$$

And the minimum value of F_w is

$$\min F_w = \frac{2D_1D_2E_{12} - D_1^2E_{22} - D_2^2E_{11}}{E_{11}E_{22} - E_{12}^2} + \int_I [W(s)]^2 ds. \tag{2.7}$$

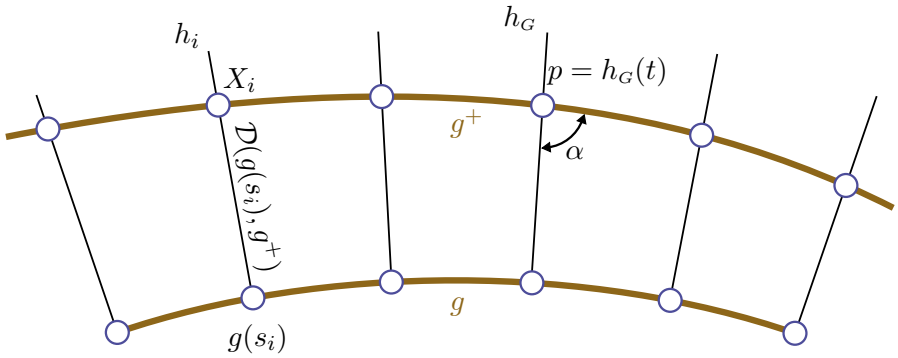


Figure 2.6: Geodesic g^+ is determined by the arc-length parameter t of a point p on h_G, and the intersection angle α between g^+ and h_G.

2.3.3 Computing the Next Geodesic

Once a Jacobi field $w(s)$ is selected, we can compute a neighboring geodesic g^+ according to $w(s)$. We uniformly sample $g(s)$ at arc-length parameter values $s_i (i = 0, \ldots, N)$, and search for a geodesic g^+ which minimizes the error function

$$E(g^+) = \sum_{i=0}^{N} [\mathcal{D}(g(s_i), g^+) - w(s_i)]^2, \tag{2.8}$$

where $\mathcal{D}(g(s_i), g^+)$ is the signed distance from $g(s_i)$ to g^+ defined in Definition 2.1. A geodesic g^+ can be determined by two degrees of freedom. We can compute g^+ either from initial value conditions (a point q on g^+ and the tangent vector of g^+ at q), or from boundary value conditions (two points on g^+). Since computing a geodesic as a boundary value problem is of higher time complexity than as an initial value problem (see Section 2.1 for a review), we determine g^+ from initial value conditions as follows. For each sample point $g(s_i)$, let h_i be the orthogonal geodesic through $g(s_i)$ which is used in computing $\mathcal{D}(g(s_i), g^+)$ (see Definition 2.1). We choose one of $\{h_i\}$ as the *generating geodesic*, and denote it by h_G. The geodesic g^+ is determined from the intersection point p between g^+ and h_G, as well as the tangent vector of g^+ at p. p can be represented using an arc-length parameter t on h_G : $p = h_G(t)$; and the tangent vector of g^+ at p can be determined from the intersection angle α between h_G and g^+ (see Figure 2.6). Therefore, the minimization of (2.8) is a non-linear least squares problem with respect to variables t and α.

We solve this minimization problem numerically using the Levenberg-Marquardt (LM) method [Madsen et al. 2004]. In each iteration, the LM method requires the partial derivatives of $\mathcal{D}(g(s_i), g^+)$ with respect to t and α. These partial derivative values are computed analytically as follows. In general, a geodesic on a triangular mesh surface lies on a sequence of faces. These faces can be "unfolded" into a planar domain. In other words, there is an isometric mapping which maps these mesh faces into a set of triangles in $\mathbb{R}^2$ with the same connectivity. And the geodesic curves on these faces are mapped to straight lines [Mitchell et al. 1987]. Figure 2.7 shows the image straight lines of g^+ and

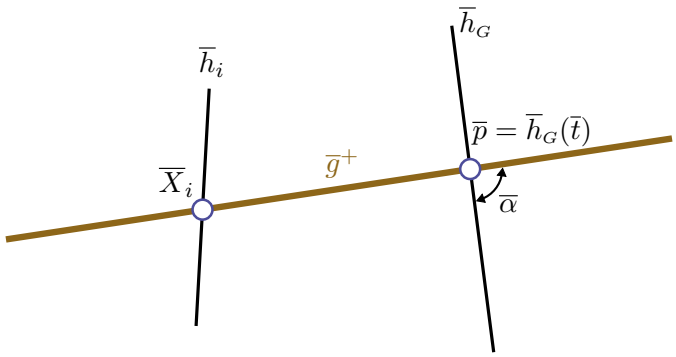


Figure 2.7: Through the unfolding of mesh faces containing g^+, geodesics g^+ and $\{h_i\}$ are mapped to straight lines $\overline{g}^+$ and $\{\overline{h}_i\}$.

$\{h_i\}$ under such a mapping. Since isometry preserves arc-length and angles, the partial derivative values can be computed using these straight lines. For the initial values of s_G and α_G, we choose

$$\begin{aligned} t^{(0)} &= t_{h_G}(p) + w(t_g(p)), \\ \alpha^{(0)} &= \tfrac{\pi}{2} + \arctan w'(t_g(p)). \end{aligned} \tag{2.9}$$

Here w is the selected Jacobi field function. $t_{h_G}(p)$ is the arc-length parameter of point p on h_G, where p is the intersection point between h_G and g. $t_g(p)$ is the arc-length parameter of p on g.

2.3.4 Evolution of Geodesics

From the neighboring geodesic g^+ computed above, we can apply the same methods to select a Jacobi field on g^+ and compute its neighboring geodesic. Repeating this procedure on each newly computed geodesic, we obtain a sequence of geodesics covering the surface region on one side of the source geodesic g. The region at the other side of g can be covered in the same way.

2.3.5 Implementation Details

Sampling of Geodesics

The sample parameters required for solving the Jacobi equation are uniformly sampled from the considered interval of g. The same sample parameters are also used for tracing orthogonal geodesics $\{h_i\}$ when computing g^+.

Computation of Gaussian Curvature

Solving the Jacobi equation requires the Gaussian curvature values at the sample points of geodesic g. To obtain these values, the input mesh surface is first pre-processed to obtain the Gaussian curvature value at each vertex, using the jet fitting algoritm [Cazals and Pouget 2008] implemented in CGAL [Pouget and Cazals 2011]. Then the Gaussian

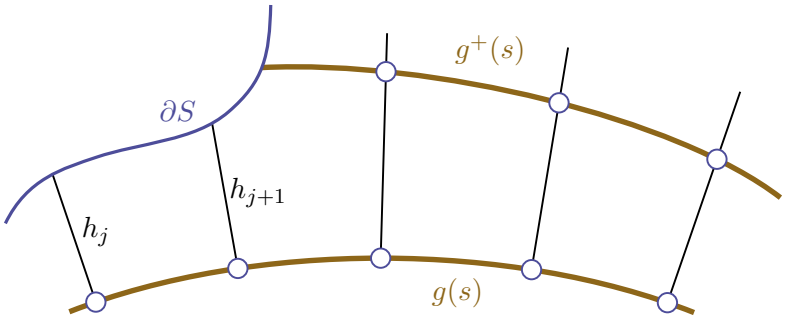


Figure 2.8: Orthogonal geodesics h_j, h_{j+1} do not intersect with $g^+(s)$, because of the surface boundary ∂S.

curvature value on each face is computed by linear interpolation of the values at the three vertices.

Intersections Between Geodesics

A neighboring geodesic g^+ may not intersect all orthogonal geodesics $\{h_i\}$ traced from the current geodesic g, especially in surface regions near the boundary. Figure 2.8 shows such an example. If h_i does not intersect with g^+, we cannot compute $\mathcal{D}(g(s_i), g^+)$ in Equation (2.8). Therefore, the target function in (2.8) is modified to the following form

$$E(t, \alpha) = \sum_{i \in \mathcal{I}} [\mathcal{D}(g(s_i), g^+) - w(s_i)]^2. \tag{2.10}$$

Here $\mathcal{I}$ is an index set such that for any $i \in \mathcal{I}$, h_i intersects with g^+. During the numerical minimization of $E(t, \alpha)$, we keep track of an index set $\mathcal{I}$ which is used for Equation (2.10). At the start of the minimization, $\mathcal{I}$ is initialized as $\mathcal{I} = \{i \mid h_i \text{ intersects } g^+_{(0)}\}$, where $g^+_{(0)}$ is the geodesic computed from initial variable values $t^{(0)}$ and $\alpha^{(0)}$ given in Equation (2.9). Besides, h_i may have more than one intersection point with g^+. To compute $\mathcal{D}(g(s_i), g^+)$, we use the intersection point which gives a signed distance value closest to $w(s_i)$.

In each iteration of the LM method, a step $(\delta t, \delta\alpha)$ is computed from the current variable values $t^{(k)}$ and $\alpha^{(k)}$, which gives new variable values $t_{\text{new}} = t^{(k)} + \delta t$, $\alpha_{\text{new}} = \alpha^{(k)} + \delta\alpha$. The classical implementation of the LM method performs a step control to determine whether the new variables are accepted: The new target function value $E(t_{\text{new}}, \alpha_{\text{new}})$ is compared with the current value $E(t^{(k)}, \alpha^{(k)})$. If $E(t_{\text{new}}, \alpha_{\text{new}}) < E(t^{(k)}, \alpha^{(k)})$, then t_{new}, α_{new} are accepted as the variable values for the next iteration $t^{(k+1)} = t_{\text{new}}$, $\alpha^{(k+1)} = \alpha_{\text{new}}$; otherwise, t_{new} and α_{new} are rejected, and another step is computed [Madsen et al. 2004]. For our problem, the new variables may give a geodesic curve g^+_{new} which does not intersect all the orthogonal geodesics indicated by the current index set $\mathcal{I}$ (see Figure 2.9 for an example). In order to compare $E(t_{\text{new}}, \alpha_{\text{new}})$ with $E(t^{(k)}, \alpha^{(k)})$, we compute a common index set $\hat{\mathcal{I}} = \mathcal{I} \cap \mathcal{I}_{\text{new}}$ where $\mathcal{I}_{\text{new}} = \{i \mid h_i \text{ intersects } g^+_{\text{new}}\}$. Then $E(t_{\text{new}}, \alpha_{\text{new}})$,

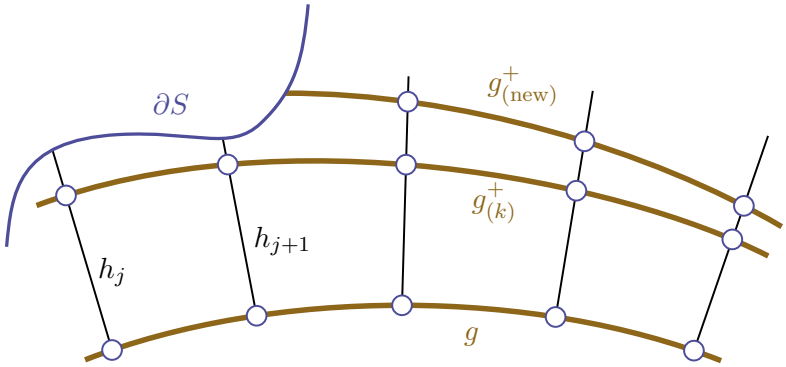


Figure 2.9: Orthogonal geodesics h_j, h_{j+1} intersect with geodesic $g^+_{(k)}$ which corresponds to the current variable values, but do not intersect with geodesic $g^+_{(\text{new})}$ which corresponds to the new variable values.

Algorithm 2.1: Modified step control for the LM method.

Input: A set of orthogonal geodesics $\{h_i\}$, the index set $\mathcal{I}$ of $\{h_i\}$ for computing the target function, current variable values t and α, current step $(\delta t, \delta\alpha)$, and step size thresholds ε_t, ε_α.

Output: Updated variable values t, α, updated index set $\mathcal{I}$, and a boolean variable *accept* representing whether the current step is accepted.

$t_{\text{new}} = t + \delta t, \quad \alpha_{\text{new}} = \alpha + \delta\alpha$;
compute geodesic g^+_{new} from t_{new} and α_{new} ;
$\mathcal{I}_{\text{new}} = \{i \mid h_i \text{ intersects } g^+_{\text{new}}\}$;
$\hat{\mathcal{I}} = \mathcal{I} \cap \mathcal{I}_{\text{new}}$;
if $\hat{\mathcal{I}} \neq \mathcal{I} \ \bigwedge \ (\ |\delta t| > \varepsilon_t \ \bigvee \ |\delta\alpha| > \varepsilon_\alpha \ )$ **then**
    accept = FALSE;
    return;
else
    compute $E(t, \alpha)$ and $E(t_{\text{new}}, \alpha_{\text{new}})$ using index set $\hat{\mathcal{I}}$;
    if $E(t_{new}, \alpha_{new}) < E(t, \alpha)$ **then**
        $t = t_{\text{new}}, \quad \alpha = \alpha_{\text{new}}, \quad \mathcal{I} = \hat{\mathcal{I}}$;
        accept = TRUE;
        return;
    else
        accept = FALSE;
        return;
    end
end

Algorithm 2.2: Searching for the longest interval of Jacobi field with bounded deviation.

Input: A set of uniform sample parameters $\{x_i | 0 \le i \le M\}$ on geodesic g, target constant distance W, and deviation threshold ε.

Output: The longest interval $[x_j, x_k]$ $(0 \le j < k \le M)$ on which a Jacobi field satisfying condition (2.12) exists.

```
for K = M to 2 do
    create empty set A ;
    for I = 0 to M − K do
        select a Jacobi field w(s) to W on interval [x_I, x_{I+K}];
        if w(s) satisfy condition (2.12) on [x_I, x_{I+K}] then
            compute fitting error E_I according to Equation (2.7);
            add index I to A;
        end
    end
    if A is not empty then
        find index Ī in A with the smallest error E_Ī;
        x_j = x_Ī,  x_k = x_{Ī+K} ;
        return;
    end
end
```

$E(t^{(k)}, \alpha^{(k)})$ are computed using $\hat{\mathcal{I}}$ instead of $\mathcal{I}$. If $E(t_{\text{new}}, \alpha_{\text{new}}) < E(t^{(k)}, \alpha^{(k)})$ then $\mathcal{I}$ is updated to $\hat{\mathcal{I}}$. On the other hand, if $\hat{\mathcal{I}} \neq \mathcal{I}$, updating $\mathcal{I}$ will lead to a smaller number of orthogonal geodesics used for computing the target function. For better fitting quality, we would like $\mathcal{I}$ to be as large as possible. Since a large step $(\delta t, \delta\alpha)$ will easily induce a geodesic g^{+}_{new} with $\hat{\mathcal{I}}$ different from $\mathcal{I}$, we reject a step if $\hat{\mathcal{I}} \neq \mathcal{I}$, and $|\delta t|$ or $|\delta\alpha|$ is not small enough. This modified step control method is summarized in Algorithm 2.1.

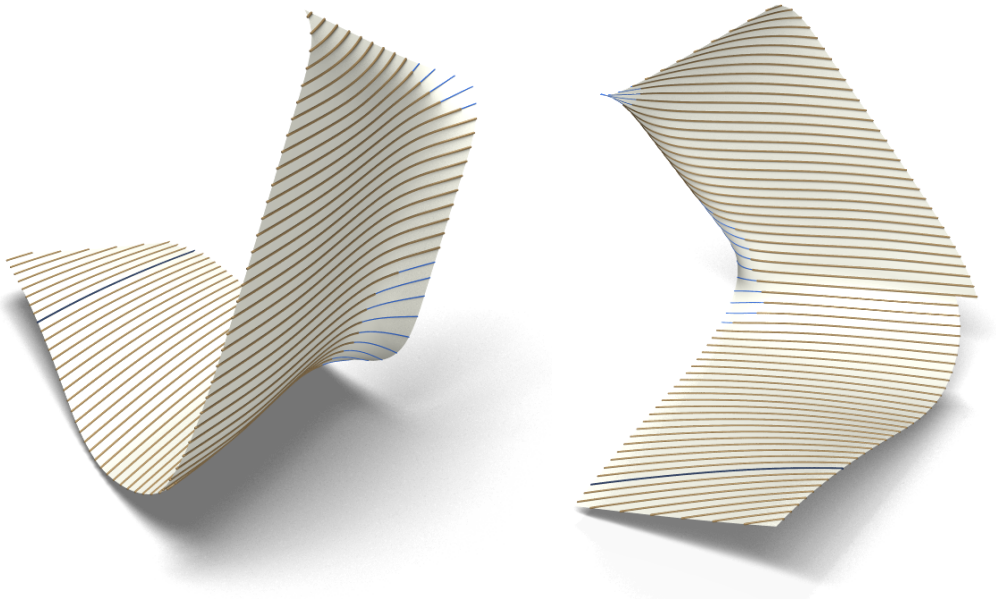
2.3.6 Results

Here we give some examples of evolution of geodesics . In all examples, the source curve is a straightest geodesic computed from a point on the surface, and a surface tangent at the point. These initial conditions are specified interactively.

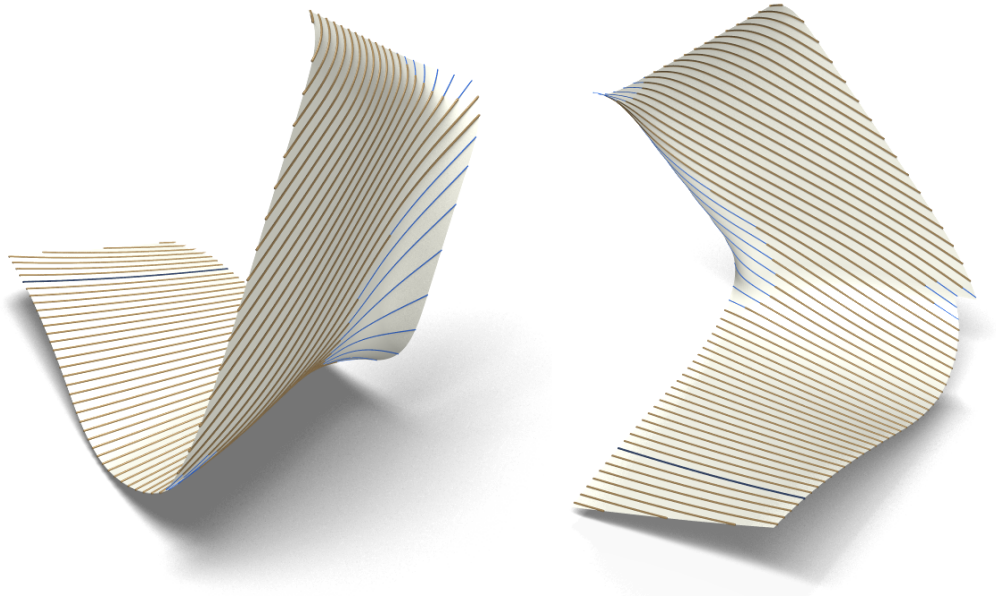
Constant Distance with Bounded Deviation

The most simple target distance function $W(s)$ is a constant function. In general, the distance function between two neighboring geodesics cannot be constant unless the surface is developable. In the evolution of geodesics, we can look for the longest interval $[a, b]$ on the current geodesic g such that there exists a Jacobi field which satisfies the condition

$$W - \varepsilon \le w(s) \le W + \varepsilon, \quad \forall s \in [a, b], \tag{2.11}$$



(a)



(b)

Figure 2.10: Evolution of geodesics with constant target distance and bounded deviation, using two different source curves (dark blue) on a same surface. Here the light blue pieces of curves show where the distance constraint (2.11) is violated. Both examples use the same target distance value W and deviation error bound $\varepsilon = 0.05 \cdot W$.

where $\varepsilon > 0$ is a threshold for deviation. In other words, we look for the longest interval with a Jacobi field of bounded deviation. To do so, we first sample $g(s)$ uniformly at arc-length parameter values $x_i = \frac{i}{M}L, i = 0, \ldots, M$, where L is the total length of $g(s)$. Among them we search for the longest interval $[x_j, x_k]$ $(0 \leq j < k \leq M)$ on which there exists a solution $w(s)$ of the Jacobi equation with the condition

$$W - \varepsilon \leq w(s_i) \leq W + \varepsilon, \quad i = j, \ldots, k. \tag{2.12}$$

Note that this condition is a discrete version of (2.11). The search is done with Algorithm 2.2. Then the Jacobi field closest to W on $[x_j, x_k]$ is selected, and a neighboring geodesic g^+ is computed using orthogonal geodesics $\{h_i\}$ traced from points on $g(s)$ within the interval $[x_j, x_k]$. Afterward the actual distance from g to g^+ needs to be checked: We trace another set of orthogonal geodesics $\{\hat{h}_i\}$ from sample points $\{g(\hat{s}_i)\}$ across the whole curve $g(s)$, and compute the signed distance values $\mathcal{D}(g(\hat{s}_i), g^+)$. Then we search for the longest interval $[\hat{s}_j, \hat{s}_k]$ such that $W - \varepsilon \leq \mathcal{D}(g(\hat{s}_i), g^+) \leq W + \varepsilon$ for $j \leq i \leq k$. And g^+ is trimmed by $\hat{h}_j$ and $\hat{h}_k$ to produce a segment $\hat{g}^+$, which represents the longest range on g^+ with approximately constant distance W from $g(s)$. Figure 2.10 shows such an example, where geodesics are evolved from two different source curves. Here the source geodesic is shown in dark blue, the segments $\hat{g}^+$ are shown in brown, and the other parts of g^+ are shown in light blue. Note that the brown curves indicate a surface region which can be cladded with straight panels of constant width W along the geodesics, without large gaps between the panels.

Curvature-dependent Evolution

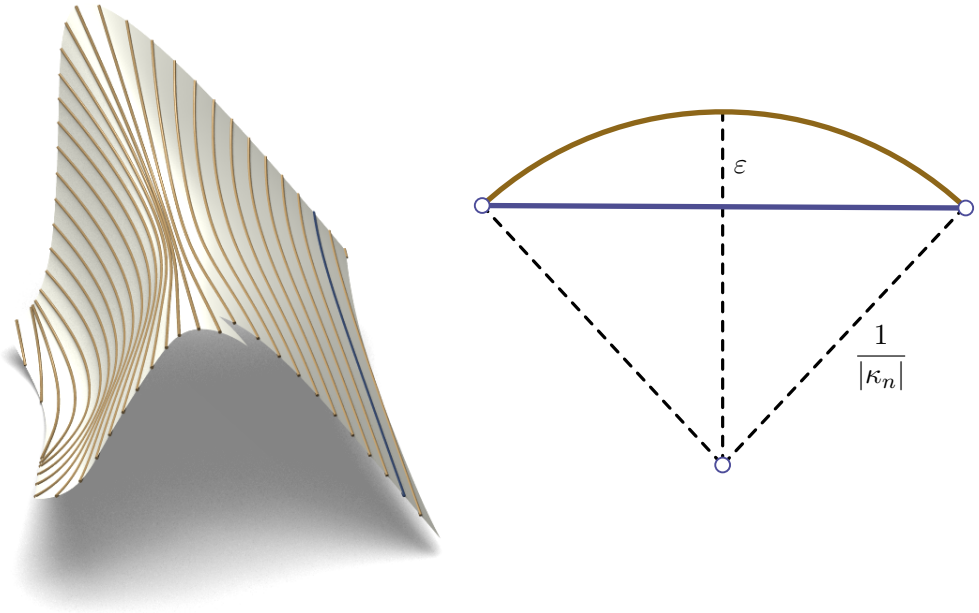
For aesthetic reasons, we might want to have denser strips if the normal curvature across the strip is high. This can be achieved with a curvature-dependent target distance function $W(s) = \phi(|\kappa_n(s)|)$, where $\kappa_n(s)$ is the normal curvature of the surface at point $g(s)$ along the direction orthogonal to $g'(s)$, and ϕ is a monotonically decreasing function. Figure 2.11(a) shows an example of using curvature-dependent target functions. Here we use function

$$\phi(|\kappa_n|) = \frac{2}{|\kappa_n|} \arcsin \sqrt{2\varepsilon|\kappa_n| - (\varepsilon|\kappa_n|)^2},$$

which is the maximum length of a circular arc of curvature $|\kappa_n|$, which deviates no more than ε from the corresponding chord, if $\varepsilon|\kappa_n| < 1$ (see Figure 2.11(b)). This circle approximates the surface sectional curve, and the chord approximates the sectional curve of a panel with boundary curves on the surface. In practice we want to have a lower bound W_U and W_L for function $W(s)$. So $W(s)$ is defined as

$$W(s) = \begin{cases} W_L & \text{if } |\kappa_n(s)| > \kappa_L, \\ W_U & \text{if } |\kappa_n(s)| < \kappa_U, \\ \phi(|\kappa_n(s)|) & \text{otherwise,} \end{cases}$$

where $\phi(\kappa_L) = W_L$, $\phi(\kappa_U) = W_U$.



(a) (b)

Figure 2.11: Curvature-dependent evolution of geodesics. (a) A family of geodesics generated from a curvature-dependent target distance function. (b) The curvature-dependent function is derived from a bound ε of the deviation between an arc and a chord on a circle of radius $1/|\kappa_n|$.

Pattern Transfer

We can even transfer straight line patterns in a plane onto geodesic patterns on a surface. First we sketch a set of straight line segments and designate one of them to be the source segment l_0. Note that straight lines are geodesics in a plane, and Definition 2.1 defines the distance function from a segment to the line containing its neighbor segment, which is a linear function. Therefore, on the two sides of l_0, we have two sets of distance functions $\{W_0^+(s), W_1^+(s), \ldots, W_{N_+}^+(s)\}$ and $\{W_0^-(s), W_1^-(s), \ldots, W_{N_-}^-(s)\}$ between neighboring straight line segments (see Figure 2.12). From a source geodesic g^0 on a surface, we can evolve a series of geodesics g^i $(i \geq 1)$ on each side of g^0, according to one of the above two sets of distance functions. For a set of distance functions $\{W_0(s), W_1(s), \ldots, W_N(s)\}$, suppose function $W_j(s)$ is measured from a segment of length $\mathcal{L}_j$ to its neighboring segment. Then geodesic g^i is computed from g^{i-1} according to a target distance function which is a "normalized" version of $W_{\mathcal{N}(i)}(s)$ where $\mathcal{N}(i) = (i-1) \bmod (N+1)$:

$$W(s) = \frac{(L_{i-1} - s)W_{\mathcal{N}(i)}(0) + sW_{\mathcal{N}(i)}(\mathcal{L}_{\mathcal{N}(i)})}{\mathcal{L}_{\mathcal{N}(i)}}.$$

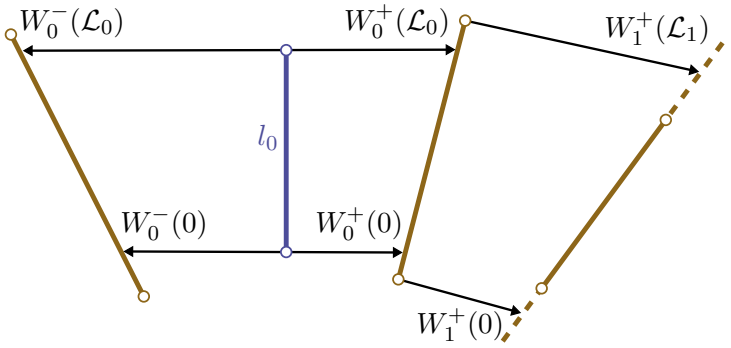
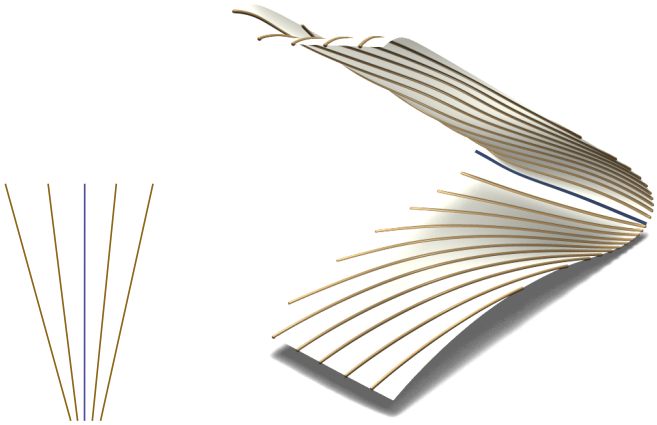
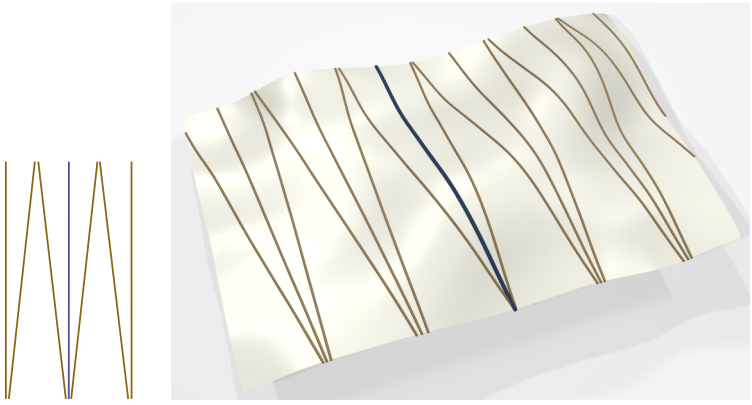


Figure 2.12: A set of straight line segments with a designated source l_0. They define two sets of linear distance functions $\{W_i^+(s)\}$ and $\{W_j^-(s)\}$ on two sides of l_0.



(a)



(b)

Figure 2.13: Evolution of geodesics according to distance functions defined by a set of straight line segments.

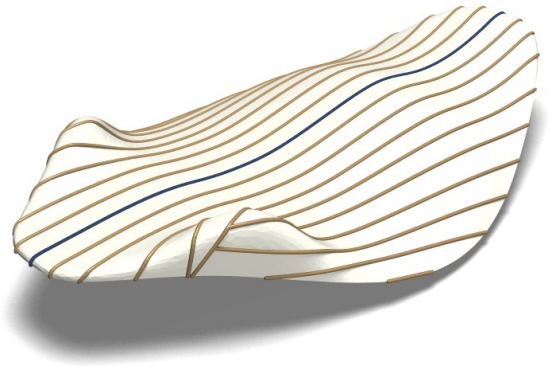


Figure 2.14: As-equidistant-as-possible evolution of geodesics on a surface. On high curvature regions, there are intersections between the geodesics. The surface is taken from London Aquatics Centre, by Zaha Hadid Architects.

Here L_{i-1} is the length of geodesic g^{i-1}. Figure 2.13 shows some straight line segment patterns in a plane, and their corresponding geodesic patterns on a surface.

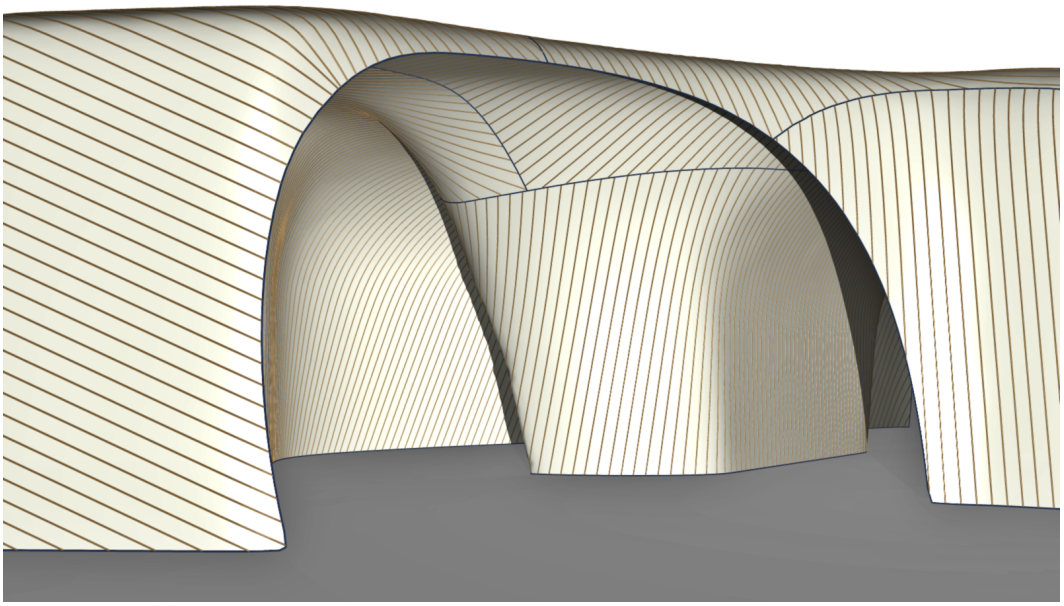
As-equidistant-as-possible Evolution

Instead of finding the longest interval of Jacobi field satisfying a deviation bound, we can simply select a Jacobi field $w(s)$ which is closest to a constant function on the whole interval of a geodesic. Depending on the underlying surface shape, the next geodesic computed from $w(s)$ may be far from being equidistant to the current one. It may even intersect the current geodesic. During evolution, we can check for intersections and trim the newly computed geodesic with existing geodesics. Such evolution gives a layout of geodesics which are "as equidistant as possible". Figure 2.14 shows an example of this construction.

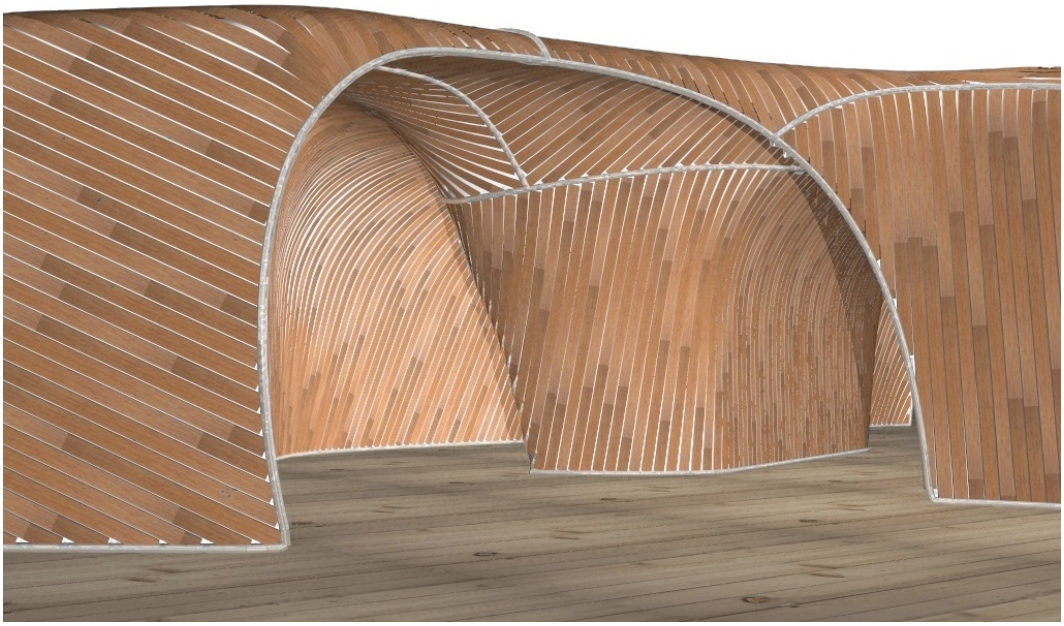
To reduce the number of intersections of geodesics, we can first segment the surface into regions each of which can be covered with geodesics without many intersections. Such a segmentation can be done using *geodesic vector fields* introduced by [Pottmann et al. 2010]. Afterward we perform as-equidistant-as-possible evolution on each region separately. Figure 2.15 shows such an example, where the generated geodesics are used to guide the cladding with straight wooden panels.

Timings

The run time of an evolution depends on the complexity of the mesh, the sample density of g for computing the Jacobi field and for tracing orthogonal geodesics, and the number of geodesics that are evolved from the source curve. Table 3.1 gives the timing data of some examples above. Here #Face is the number of faces of the surface mesh. δs is the ratio of the distance between neighboring sample points along g to the bounding box diameter of the surface patch. Time is the total run time of the evolution in seconds,



(a)



(b)

Figure 2.15: As-equidistant-as-possible evolution on a segmented surface. (a) Segmentation boundary (in blue), and geodesics (in brown) within each region. (b) Design of wooden panel patterns based on (a).

measured on a laptop with a 2.4GHz dual core CPU. %Jacobi and %NLS are the percentages of the total time used in computing Jacobi fields and in solving the non-linear least squares problem for computing g^+, respectively. We can see that most of the time is spent on computing g^+ by least squares minimization.

Fig	#Face	δs	Time	%Jacobi	%NLS
2.10(a)	95594	$6.4 \cdot 10^{-4}$	13.4	0.82	99.17
2.10(b)	95594	$6.4 \cdot 10^{-4}$	12.1	0.91	98.84
2.11	205886	$4.3 \cdot 10^{-4}$	29.2	0.58	98.66
2.13(a)	164699	$4.1 \cdot 10^{-4}$	8.7	0.57	99.42
2.13(b)	49632	$9.7 \cdot 10^{-4}$	3.5	0.56	99.15

Table 2.1: Timing data for evolution of geodesics.

2.3.7 Limitations of Geodesic Evolution Method

Figure 2.14 shows an example where neighboring geodesics intersect, which may be undesirable in applications. We can avoid such intersections only if we can find a solution of the Jacobi equation with no zeros in the considered interval. It turns out that we can tell the existence of such "useful" solutions simply by testing if the fundamental solution $w^{(1)}$ with initial conditions $w_0^{(1)} = 0$, $w_1^{(1)} = 1$ has another zero in the interval. The proposition below, proved in [do Carmo 1992], characterizes the two possible cases and sums up some of their geometric properties.

Proposition 2.1. *Consider a geodesic curve $g(s)$, where $s \in [0, L]$ is an arc length parameter. Assume a fundamental solution $w^{(1)}(s)$ of the Jacobi equation with $w^{(1)}(0) = 0$. Then there are the following two cases:*

	case 1	*case 2*
number of zeros of $w^{(1)}(s)$ for $0 < s \leq L$	*0*	*≥ 1*
number of zeros of any solution on interval $[0, 1]$	*≤ 1*	*≥ 1*
existence of solution nonzero in $[0, L]$	*yes*	*no*
g locally minimizes distance of $g(0), g(L)$	*yes*	*no*

The inequality $K(s) > (\pi/L)^2$ for $s \in [0, L]$ implies case 2. Analogously $K(s) < (\pi/L)^2$ implies case 1.

Here case 1 corresponds to the situation of no intersection. Proposition 2.1 means that on surface regions with large positive Gaussian curvature, we cannot avoid intersections if the geodesics are long enough. In these regions, we can either perform segmentation, like Figure 2.15, or consider *piecewise geodesic curves* introduced in Section 2.4.

2.4 Evolving Piecewise Geodesic Curves

2.4.1 Piecewise Geodesic Curves

On surface regions with large Gaussian curvature, the Jacobi equation implies that the distance function $w(s)$ between two neighboring geodesics will quickly deviate from a constant function. In other words, only on a short interval can $w(s)$ satisfy the condition $W-\varepsilon \leq w(s) \leq W+\varepsilon$ with constant W and small ε. This motivates us to cover surfaces using curves consisting of short geodesic segments, in order to maintain approximately constant distance between neighboring curves. We call such curves *piecewise geodesic curves*.

Definition 2.2. *A piecewise geodesic curve is a continuous curve consisting of geodesic segments. If $g(s), s \in [0, L]$ is the arc-length parametrization of a piecewise geodesic curve, then there exists a set of parameter values $0 = s_0 < s_1 < \ldots < s_N = L$ such that $g(s)$ is geodesic on intervals $[s_i, s_{i+1}], i = 0, \ldots, N-1$. Each segment $\{g(s) \mid s \in [s_i, s_{i+1}]\}$ is called an* arc *of g, and is denoted by g_i. The points $g(s_i)(i = 1, \ldots, N-1)$ are called the* breakpoints *of g.*

Note that the tangent vector of a piecewise geodesic curve may have discontinuity at the breakpoints. Also note that a geodesic curve is a special piecewise geodesic curve (consisting of only one segment of geodesic).

2.4.2 Jacobi Fields on Piecewise Geodesic Curves

Like geodesic curves, we can employ Jacobi fields to approximate the distance between two neighboring piecewise geodesic curves. For a piecewise geodesic curve $g(s)$, we consider an infinitesimally close piecewise geodesic curve $g^+(s)$ with the same number of breakpoints. For an arc g_i of $g(s)$, the distance function to the corresponding arc g_i^+ of $g^+(s)$ is defined as the distance function from g_i to the geodesic curve containing g_i^+, using Definition 2.1. This distance function can be approximated by a solution function $w_i(s)$ of the Jacobi equation on g_i. If each arc g_i corresponds to interval $[s_i, s_{i+1}]$ of $g(s)$, and each arc g_i^+ corresponds to interval $[s_i^+, s_{i+1}^+]$ of $g^+(s)$, then infinitesimally we have the following condition for functions $w_{i-1}(s)$, $w_i(s)$ on neighboring arcs g_{i-1}, g_i (see Figure 2.16)

$$\frac{w_{i-1}(s_i)}{\cos\alpha_i} = \frac{w_i(s_i)}{\cos\beta_i}. \tag{2.13}$$

Here breakpoints $g(s_i)$ and $g^+(s_i^+)$ are connected by a geodesic h_i. α_i, β_i are the angles between the tangent of h_i at $g(s_i)$ and directions $R_{\pi/2}\left(g'_{i-1}(s_i)\right)$, $R_{\pi/2}\left(g'_i(s_i)\right)$, respectively.

2.4.3 Selecting Jacobi Fields on Piecewise Geodesic Curves

Given a target distance function $W(s)$ and a piecewise geodesic curve $g(s)$ with arcs g_i defined on intervals $[s_i, s_{i+1}]$ $(i = 0, \ldots, N-1)$, we can select a set of Jacobi fields $\{w_i(s)\}$, one for each arc, to approximate $W(s)$. First we specify the directions of movement for

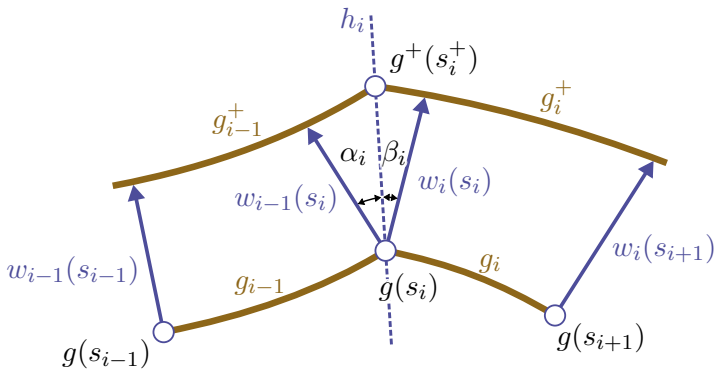


Figure 2.16: Jacobi fields on a piecewise geodesic curve $g(s)$.

the breakpoints, from which we read off the angles α_i, β_i in relation (2.13). Then $\{w_i(s)\}$ are obtained by minimizing a target function

$$E = \sum_{i=0}^{N-1} \int_{s_i}^{s_{i+1}} (w_i(s) - W(s))^2 ds$$

subject to constraints (2.13) for $i = 1, \ldots, N-1$. On $[s_i, s_{i+1}]$, $w_i(s)$ can be represented as linear combination of two fundamental solutions $w^{(i,1)}(s)$, $w^{(i,2)}(s)$

$$w_i(s) = \lambda_i^{(1)} w^{(i,1)}(s) + \lambda_i^{(2)} w^{(i,2)}(s) \tag{2.14}$$

with boundary conditions $w^{(i,1)}(s_i) = 1$, $w^{(i,1)}(s_{i+1}) = 0$, $w^{(i,2)}(s_i) = 0$, $w^{(i,2)}(s_{i+1}) = 1$. To satisfy condition (2.13), we introduce coefficient variables $\lambda_i (i = 0, \ldots, N)$, and let

$$w_i(s) = \begin{cases} \lambda_i w^{(i,1)}(s) + \lambda_{i+1} w^{(i,2)}(s), & i = 0, \\ \frac{\cos\beta_i}{\cos\alpha_i} \lambda_i w^{(i,1)}(s) + \lambda_{i+1} w^{(i,2)}(s), & i = 1, \ldots, N-1. \end{cases} \tag{2.15}$$

Now the target function can be written in matrix form

$$E = \boldsymbol{\lambda}^T \mathbf{A} \boldsymbol{\lambda} - 2\boldsymbol{\lambda}^T \mathbf{B} + \int_{s_0}^{s_N} [W(s)]^2 ds.$$

Here

$$\boldsymbol{\lambda} = [\lambda_0, \ldots, \lambda_N]^T,$$

$$\mathbf{A} = \begin{bmatrix} H_0^{11} & H_0^{12} & & & & & & & \\ H_0^{12} & H_0^{22}+(\gamma_1)^2 H_1^{11} & \gamma_1 H_1^{12} & & & & & & \\ & \gamma_1 H_1^{12} & \ddots & & & & & & \\ & & & \gamma_{i-1} H_{i-1}^{12} & H_{i-1}^{22}+(\gamma_i)^2 H_i^{11} & \gamma_i H_i^{12} & & & \\ & & & & \gamma_i H_i^{12} & \ddots & & & \\ & & & & & \gamma_{N-2} H_{N-2}^{12} & H_{N-2}^{22}+(\gamma_{N-1})^2 H_{N-1}^{11} & \gamma_{N-1} H_{N-1}^{12} \\ & & & & & & \gamma_{N-1} H_{N-1}^{12} & H_{N-1}^{22} \end{bmatrix},$$

$$\mathbf{B} = [D_0^{(1)},\ D_0^{(2)}+\gamma_1 D_1^{(1)},\ \ldots,\ D_{i-1}^{(2)}+\gamma_i D_i^{(1)},\ \ldots,\ D_{N-2}^{(2)}+\gamma_{N-1} D_{N-1}^{(1)},\ D_{N-1}^{(2)}]^T,$$

where

$$\begin{aligned} \gamma_i &= \frac{\cos\beta_i}{\cos\alpha_i}, \\ D_i^{(k)} &= \int_{s_i}^{s_{i+1}} w^{(i,k)}(s)\ W(s)\ ds, \quad k=1,2, \\ H_i^{jk} &= \int_{s_i}^{s_{i+1}} w^{(i,j)}(s)\ w^{(i,k)}(s)\ ds, \quad j,k=1,2. \end{aligned}$$

Minimizing E leads to a symmetric tridiagonal system

$$\mathbf{A}\boldsymbol{\lambda} = \mathbf{B}.$$

And functions $\{w_i(s)\}$ are constructed from the solution to this system.

2.4.4 Handling Intervals Without Fundamental Solutions

On each interval $[s_j, s_{j+1}]$, the two fundamental solutions $w^{(j,1)}(s)$, $w^{(j,2)}(s)$ in Equation (2.14) are computed from linear system (2.6). If this system is singular, $w^{(j,1)}(s)$, $w^{(j,2)}(s)$ are not available. In this case, we add breakpoints to split the interval $[s_i, s_{i+1}]$, until we can solve system (2.6) to obtain fundamental solutions on each sub-interval of $[s_i, s_{i+1}]$. To do so, we first uniformly sample the interval at $t_k = s_j + \frac{k}{M_j}(s_{j+1} - s_j)$ $(k = 1, \ldots, M_j - 1)$ as candidate breakpoints. For each t_k, we check whether linear system (2.6) is non-singular on both intervals $[s_j, t_k]$ and $[t_k, s_{j+1}]$. If this is the case, two solution functions $w_{j,L}(s)$ and $w_{j,R}(s)$ are computed for intervals $[s_j, t_k]$ and $[t_k, s_{j+1}]$ respectively, by minimizing the target function

$$e_k = \int_{s_j}^{t_k} [w_{j,L}(s) - W(s)]^2\ ds + \int_{t_k}^{s_{j+1}} [w_{j,R}(s) - W(s)]^2\ ds. \tag{2.16}$$

$w_{j,L}(s)$ and $w_{j,R}(s)$ need to satisfy condition (2.13) at t_k, which means

$$w_{j,L}(t_k) = w_{j,R}(t_k). \tag{2.17}$$

In order to minimize e_k subject to condition (2.17). we introduce three coefficient variables λ_0, λ_1, λ_2, and let

$$\begin{aligned} w_{j,L}(s) &= \lambda_0 w_{j,L}^{(1)}(s) + \lambda_1 w_{j,L}^{(2)}(s), \\ w_{j,R}(s) &= \lambda_1 w_{j,R}^{(1)}(s) + \lambda_2 w_{j,R}^{(2)}(s), \end{aligned} \tag{2.18}$$

where $w_{j,L}^{(1)}(s)$, $w_{j,L}^{(2)}(s)$ are fundamental solutions on $[s_j, t_k]$ with boundary conditions $w_{j,L}^{(1)}(s_j) = 1$, $w_{j,L}^{(1)}(t_k) = 0$, $w_{j,L}^{(2)}(s_j) = 0$, $w_{j,L}^{(2)}(t_k) = 1$; and $w_{j,R}^{(1)}(s)$, $w_{j,R}^{(2)}(s)$ are fundamental solutions on $[t_k, s_{j+1}]$ with boundary conditions $w_{j,R}^{(1)}(t_k) = 1$, $w_{j,R}^{(1)}(s_{j+1}) = 0$,

$w_{j,R}^{(2)}(t_k) = 0$, $w_{j,R}^{(2)}(s_{j+1}) = 1$. Minimizing e_k leads to linear equations

$$\begin{aligned}
\frac{\partial e}{\partial \lambda_0} &= 2(\lambda_0 H_L^{11} + \lambda_1 H_L^{12} - D_L^{(1)}) = 0, && (2.19)\\
\frac{\partial e}{\partial \lambda_1} &= 2[\lambda_0 H_L^{12} + \lambda_1 (H_L^{22} + H_R^{11}) + \lambda_2 H_R^{12} - D_L^{(2)} - D_R^{(1)}] = 0, && (2.20)\\
\frac{\partial e}{\partial \lambda_2} &= 2(\lambda_1 H_R^{12} + \lambda_2 H_R^{22} - D_R^{(2)}) = 0, && (2.21)
\end{aligned}$$

where

$$\begin{aligned}
H_L^{ik} &= \int_{s_j}^{t_k} w_{j,L}^{(i)}(s)\ w_{j,L}^{(k)}(s)\ ds, \quad i,k = 1,2,\\
H_R^{ik} &= \int_{t_k}^{s_{j+1}} w_{j,R}^{(i)}(s)\ w_{j,R}^{(k)}(s)\ ds, \quad i,k = 1,2,\\
D_L^{(i)} &= \int_{s_j}^{t_k} w_{j,L}^{(i)}(s)\ W(s)\ ds, \quad i = 1,2,\\
D_R^{(i)} &= \int_{t_k}^{s_{j+1}} w_{j,R}^{(i)}(s)\ W(s)\ ds, \quad i = 1,2.
\end{aligned}$$

The minimum value of e_k is computed from the solution of linear system (2.19) – (2.21), and is denoted by $\mathcal{E}_k$. Among all candidate breakpoints $\{t_k\}$, if at least for one of them we can compute $\mathcal{E}_k$, then the one with the smallest value of $\mathcal{E}_k$ is chosen as breakpoint, and no more breakpoints are needed on $[s_j, s_{j+1}]$. Otherwise, we simply split $[s_j, s_{j+1}]$ at the midpoint $(s_j + s_{j+1})/2$, and repeat the above process to add breakpoints to sub-intervals $[s_j, (s_j + s_{j+1})/2]$ and $[(s_j + s_{j+1})/2, s_{j+1}]$. Proposition 2.1 guarantees that linear system (2.6) is non-singular on an interval that is short enough. Therefore, only a finite number of breakpoints are needed for interval $[s_j, s_{j+1}]$.

2.4.5 Adding Breakpoints to a Piecewise Geodesic Curve

The Jacobi fields computed in Section 2.4.3 may have large deviation from the target function $W(s)$. For each function $w_i(s)$, we check for the condition

$$W(s_{i,k}) - \varepsilon \le w_i(s_{i,k}) \le W(s_{i,k}) + \varepsilon, \quad k = 0, \ldots, M_i, \tag{2.22}$$

where $\varepsilon > 0$ is a given error threshold, and $\{s_{i,k} \mid k = 0, \ldots, M_i\}$ is a set of sample parameter values on interval $[s_i, s_{i+1}]$. If this condition is violated, a breakpoint is added to $[s_i, s_{i+1}]$, and a new set of functions $\{w_i(s)\}$ is computed using the new breakpoints. If there is more than one interval violating condition (2.22), we add the breakpoint to the interval with the largest value of the following average error function

$$\overline{e}_i = \frac{1}{s_{i+1} - s_i} \int_{s_i}^{s_{i+1}} [w_i(s) - W(s)]^2\ ds.$$

This process is repeated until condition (2.22) is satisfied on all intervals.

When adding a breakpoint to interval $[s_j, s_{j+1}]$, we determine the breakpoint position in the same way as Section 2.4.4, with the additional constraint that $w_{j,L}(s)$ and $w_{j,R}(s)$ in Equation (2.16), together with the current Jacobi fields $\{w_i(s)|i \neq j\}$ on other intervals, need to satisfy condition (2.13) at the current breakpoints. Depending on the position of interval $[s_j, s_{j+1}]$ within the whole curve, this additional constraint could mean some boundary conditions for $w_{j,L}(s)$ and $w_{j,R}(s)$ at s_j and s_{j+1}: If there exists a neighboring interval $[s_{j-1}, s_j]$, condition (2.13) at s_j means that

$$\frac{w_{j-1}(s_j)}{\cos\alpha_j} = \frac{w_{j,L}(s_j)}{\cos\beta_j} \;\Leftrightarrow\; w_{j,L}(s_j) = \frac{\cos\beta_j}{\cos\alpha_j} w_{j-1}(s_j). \tag{2.23}$$

Similarly, if there exists a neighboring interval $[s_{j+1}, s_{j+2}]$, we have a boundary condition

$$w_{j,R}(s_{j+1}) = \frac{\cos\alpha_{j+1}}{\cos\beta_{j+1}} w_{j+1}(s_{j+1}). \tag{2.24}$$

Also note that Equation (2.18) implies that $w_{j,L}(s_j) = \lambda_0$, $w_{j,R}(s_{j+1}) = \lambda_2$. Therefore, condition (2.23) is equivalent to

$$\lambda_0 = \frac{\cos\beta_j}{\cos\alpha_j} w_{j-1}(s_j),$$

and condition (2.24) is equivalent to

$$\lambda_2 = \frac{\cos\alpha_{j+1}}{\cos\beta_{j+1}} w_{j+1}(s_{j+1}).$$

For $g(s)$ with N intervals $[s_i, s_{i+1}]$ $(i = 0, \ldots, N-1)$, we are in one of the following three cases:

1. $N = 1$: No boundary condition is imposed. We solve Equations (2.19) (2.20) (2.21) for λ_0, λ_1, λ_2.

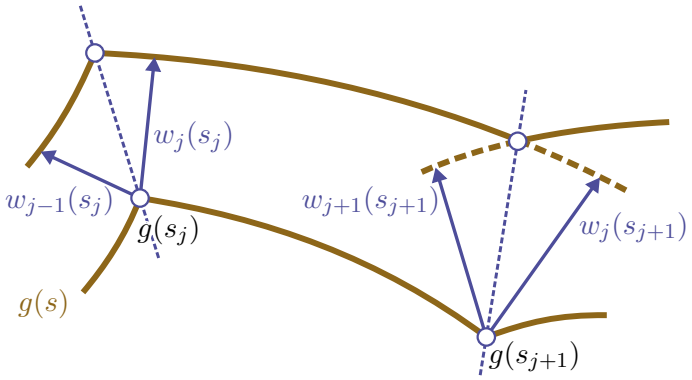
2. $N > 1, j = 0, N-1$: There is one boundary condition. For $j = 0$, λ_2 is determined from the boundary condition, and we solve Equations (2.19) (2.20) for λ_0, λ_1. For $j = 0$, the boundary condition determines λ_0, and Equations (2.20) (2.21) are solved for λ_1, λ_2.

3. $N > 1, 1 \leq j \leq N-2$: There are two boundary conditions, which determine λ_0 and λ_2. And λ_1 is determined from Equation (2.20).

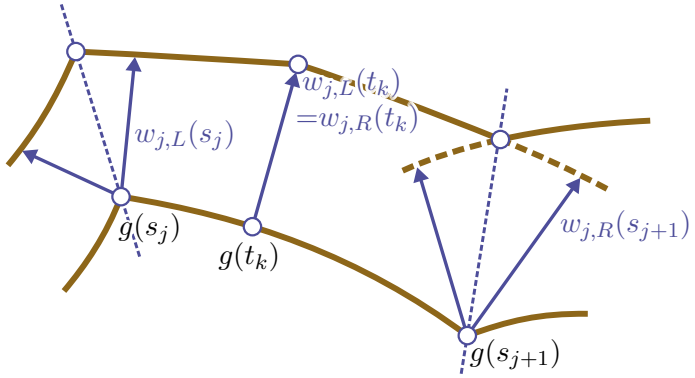
Figure 2.17 illustrates the process of adding a breakpoint.

2.4.6 Computing the Next Piecewise Geodesic Curve

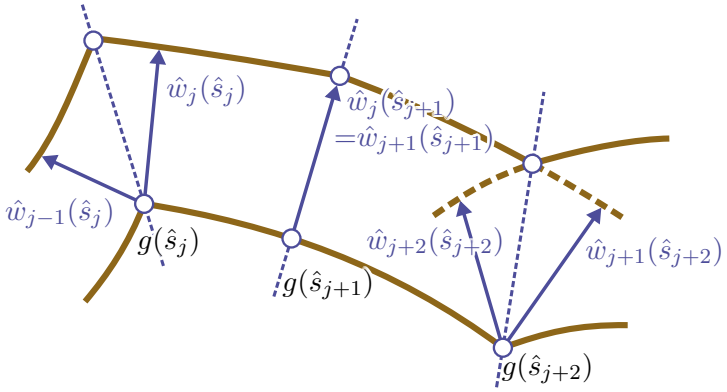
From the functions $\{w_i(s)\}$ on each arc of $g(s)$, we can compute a corresponding piecewise geodesic curve $g^+(s)$. If $g(s)$ consists of only one arc, $g^+(s)$ is a geodesic curve, and is computed in the same way as in Section 2.3.3. Otherwise, from each breakpoint P_i of



(a)



(b)



(c)

Figure 2.17: Adding a breakpoint to piecewise geodesic curve. (a) The functions $\{w_i(s)\}$ before adding a breakpoint. Interval $[s_j, s_{j+1}]$ violates condition (2.22). (b) For each candidate point t_k on $[s_j, s_{j+1}]$, functions $w_{j,L}(s)$ and $w_{j,R}(s)$ and the error term $\mathcal{E}_k$ are computed; (c) The candidate with the smallest value of $\mathcal{E}_k$ is selected as breakpoint, and functions $\{\hat{w}_i(s)\}$ are recomputed using the new breakpoints $\{g(\hat{s}_i)\}$.

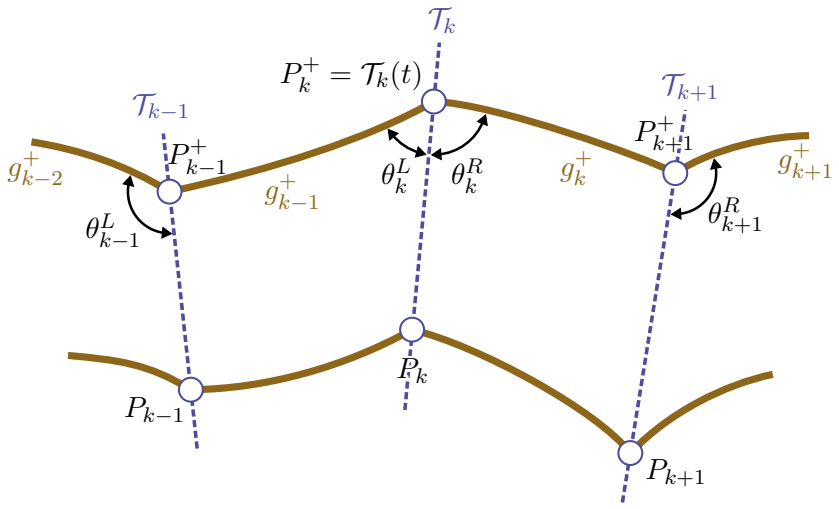


Figure 2.18: The piecewise geodesic curve $g^+(s)$ is determined by arc-length parameter t of P_k^+ on $\mathcal{T}_k$, and the angles θ_i^L $(1 \le i \le k)$, θ_j^R $(k \le j \le N-1)$.

$g(s)$, we trace a geodesic $\mathcal{T}_i$ along the movement direction of P_i which defines the relation (2.13). $\mathcal{T}_i$ is called a *trail curve*. $g^+(s)$ is constructed as a piecewise geodesic curve with the same number of breakpoints as $g(s)$. And each breakpoint P_i^+ of $g^+(s)$ needs to lie on the corresponding trail curve $\mathcal{T}_i$.

A piecewise geodesic curve $g^+(s)$ with N arcs is determined by $N+1$ degrees of freedom. We choose one trail curve $\mathcal{T}_k$ as the *main trail curve*. The break point P_k^+ of $g^+(s)$ can be determined by an arc-length parameter t on $\mathcal{T}_k$: $P_k^+ = \mathcal{T}_k(t)$. Arc g_{k-1}^+ is traced from P_k^+ along a direction given by the angle θ_k^L between g_{k-1}^+ and $\mathcal{T}_k$. g_{k-1}^+ intersects trail curve $\mathcal{T}_{k-1}$ at breakpoint P_{k-1}^+, from which we continue to trace the arcs g_{i-1}^+ $(i < k)$ one by one, using the angles θ_i^L between g_i^+ and $\mathcal{T}_i$. Similarly, starting from P_k^+, we can consecutively trace each arc g_j^+ $(j \ge k)$ using angle θ_j^R (see Figure 2.18). Therefore, $g^+(s)$ is determined by $N+1$ variables: the arc-length parameter t of P_k^+ on $\mathcal{T}_k$, and the angles $\{\theta_i^L \mid 1 \le i \le k\}$ and $\{\theta_j^R \mid k \le j \le N-1\}$.

$g^+(s)$ is obtained from a non-linear least squares problem similar to Section 2.3.3. For each arc g_i of $g(s)$ defined on interval $[s_i, s_{i+1}]$, we uniformly sample g_i at parameter values $s_{i,j} = s_i + \frac{j}{M_i}(s_{i+1} - s_i)$ $(j = 0, \ldots, M_i)$. Geodesics $h_{i,j}$ are traced from points $g(s_{i,j})$ along directions orthogonal to g_i. Signed distance values $\mathcal{D}(g(s_{i,j}), g_i^+)$ are derived from the intersections between $h_{i,j}$ and the geodesic curve containing arc g_i^+ of $g^+(s)$ (see Figure 2.19). $g^+(s)$ is determined by minimizing the following target function with respect to variables t, θ_i^L $(1 \le i \le k)$ and θ_j^R $(k \le j \le N-1)$

$$E = \sum_i \frac{s_{i+1} - s_i}{M_i + 1} \sum_{j=0}^{M_i} [\mathcal{D}(g(s_{i,j}), g_i^+) - w_i(s_{i,j})]^2. \tag{2.25}$$

This non-linear least squares problem is solved using the LM method. The partial

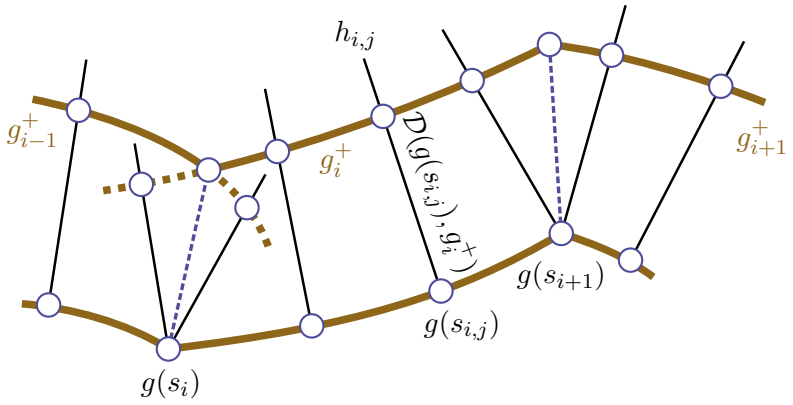


Figure 2.19: Orthogonal geodesic $h_{i,j}$ is traced from $g(s_{i,j})$ to compute signed distance $\mathcal{D}(g(s_{i,j}), g_i^+)$ to g_i^+.

derivatives of E are computed analytically by unfolding the mesh faces containing $g^+(s)$, which maps $g^+(s)$, $\{\mathcal{T}_i\}$ and $\{h_{i,j}\}$ to intersecting straight line segments (see Figure 2.20). Within the LM method, we apply modified step size control as discussed in Section 2.3.5.

2.4.7 Post-processing of the Next Piecewise Geodesic Curve

After we obtain $g^+(s)$ by numerical minimization, we still need some post-processing on $g^+(s)$ to improve its shape.

Merging Neighboring Arcs

Sometimes it turns out that some breakpoints on $g^+(s)$ are not necessary and we can replace some neighboring arcs of $g^+(s)$ with a single arc, without violating the bound

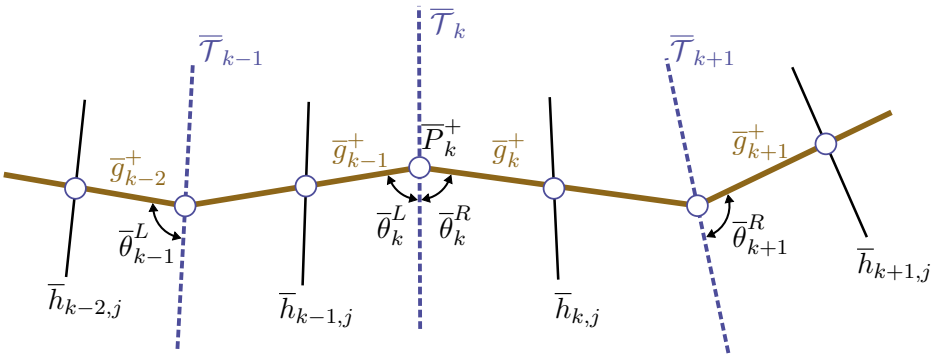


Figure 2.20: By unfolding the mesh faces containing $g^+(s)$, arcs $\{g_i^+\}$ are mapped to straight line segments $\{\overline{g}_i^+\}$, and geodesics $\{\mathcal{T}_i\}$, $\{h_{i,j}\}$ are mapped to straight lines $\{\overline{\mathcal{T}}_i\}$, $\{\overline{h}_{i,j}\}$.

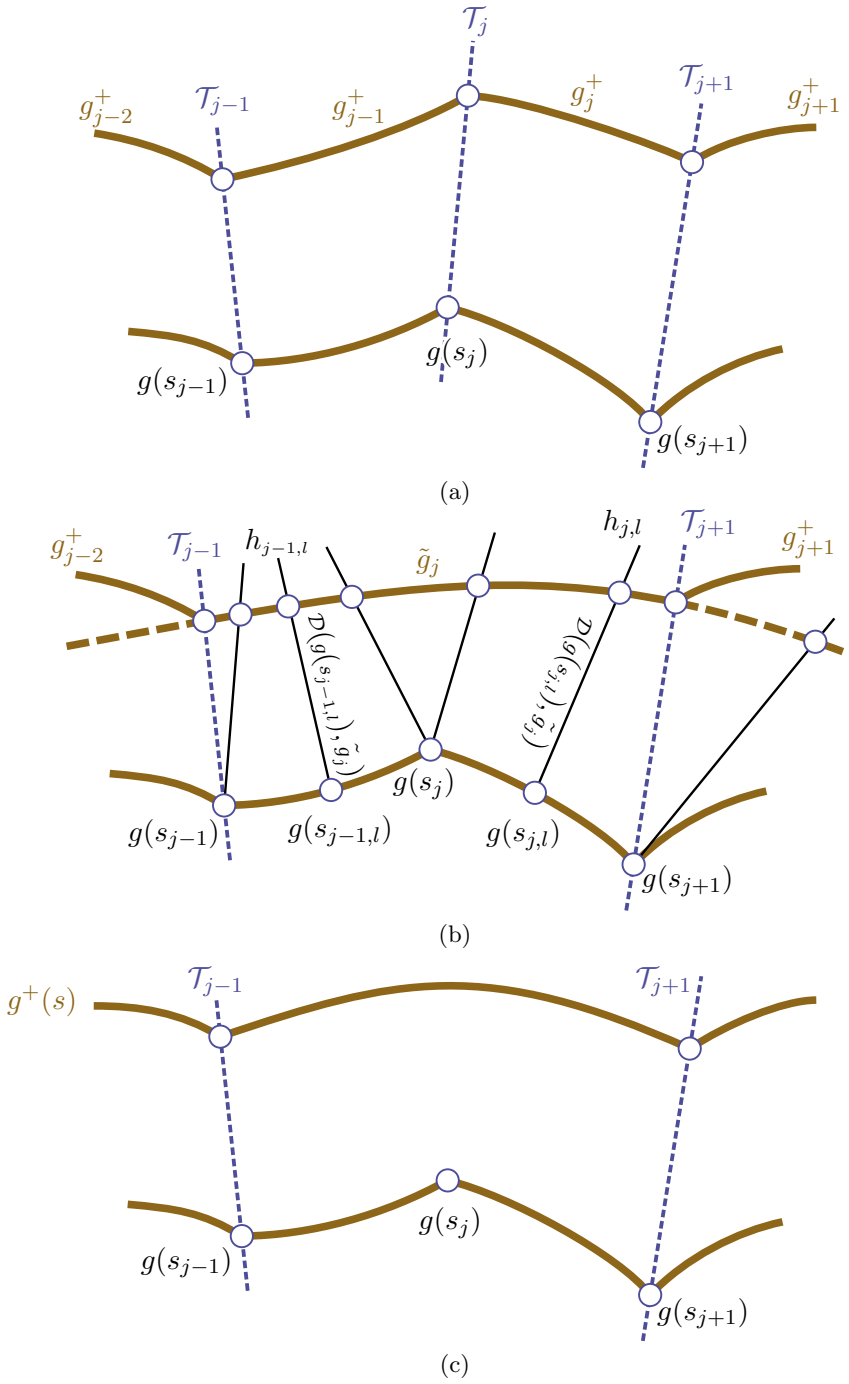


Figure 2.21: Merging neighboring arcs on $g^+(s)$. (a) The arcs before merging. (b) A replacement arc $\tilde{g}_j$ is computed for arcs g^+_{j-1} and g^+_j, and condition (2.26) is checked. (c) If condition (2.26) is satisfied, $\mathcal{T}_j$ is removed and $g^+(s)$ is recomputed using the remaining trail curves.

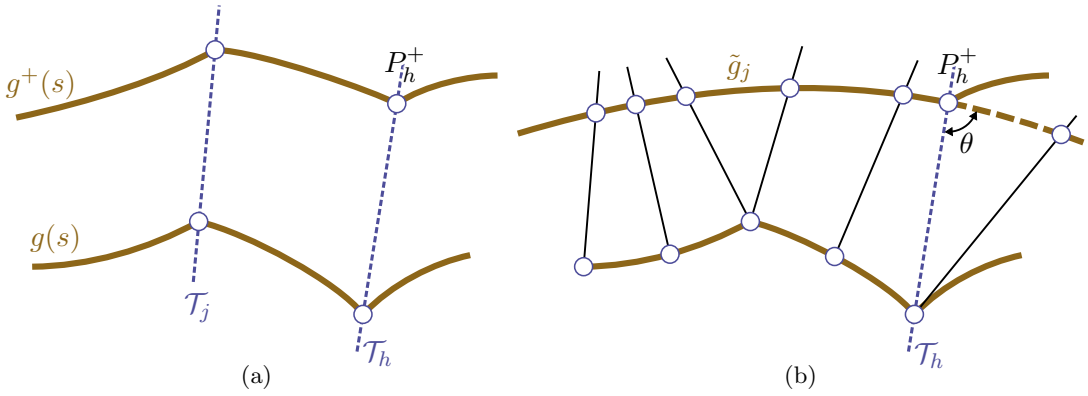


Figure 2.22: If $\mathcal{T}_j$ is the first or last trail curve, $\tilde{g}_j$ is determined by an angle θ.

of deviation between the target distance function $W(s)$ and the actual distance. This can happen because the Jacobi field is only a first order approximation of the actual distance. We would like to merge these arcs and reduce the number of breakpoints of $g^+(s)$, because this leads to a more simple and aesthetic shape of $g^+(s)$. Note that the possibility of merging two arcs can be indicated by a small intersection angle between them. Therefore, we choose a pair of neighboring arcs g_{j-1}^+ and g_j^+ with the smallest intersection angle α_j. If α_j is smaller than a given threshold ε_α, we try to merge g_{j-1}^+ and g_j^+. To do so, we first compute a replacement arc $\tilde{g}_j$ without changing the other arcs. Then with the sample parameters $\{s_{j-1,l}\}$, $\{s_{j,l}\}$ and orthogonal geodesics $\{h_{j-1,l}\}$, $\{h_{j,l}\}$ which have been used previously for computing $g^+(s)$, we check the following condition

$$W(s_{\gamma,l}) - \varepsilon \leq \mathcal{D}(g(s_{\gamma,l}), \tilde{g}_j) \leq W(s_{\gamma,l}) + \varepsilon, \quad \gamma = j-1, j, \tag{2.26}$$

where $\mathcal{D}(g(s_{\gamma,l}), \tilde{g}_j)$ is the signed distance from $g(s_{\gamma,l})$ to the geodesic curve containing $\tilde{g}_j$ (see Figure 2.21(b)), and $\varepsilon > 0$ is a threshold value. If condition (2.26) is satisfied for all sample parameters $\{s_{j-1,l}\}$, $\{s_{j,l}\}$, then trail curve $\mathcal{T}_j$ is removed, and $g^+(s)$ is recomputed using the remaining trail curves (see Figure 2.21(c)). This procedure is repeated until no neighboring arcs can be merged.

Depending on the position of $\mathcal{T}_j$, there are different ways to compute the replacement geodesic $\tilde{g}_j$:

1. $\mathcal{T}_j$ is the only trail curve: $\tilde{g}_j$ is computed by minimizing the target function

$$\tilde{E} = \sum_{\gamma=j-1,j} \frac{s_{\gamma+1} - s_\gamma}{M_\gamma + 1} \sum_{l=0}^{M_\gamma} [\mathcal{D}(g(s_{\gamma,l}), \tilde{g}_j) - W(s_{\gamma,l})]^2. \tag{2.27}$$

   This is similar to Section 2.3.3.

2. There is more than one trail curve, and $\mathcal{T}_j$ is the first or last trail curve: Since other arcs of $g^+(s)$ can not be changed, $\tilde{g}_j$ needs to be incident with the breakpoint

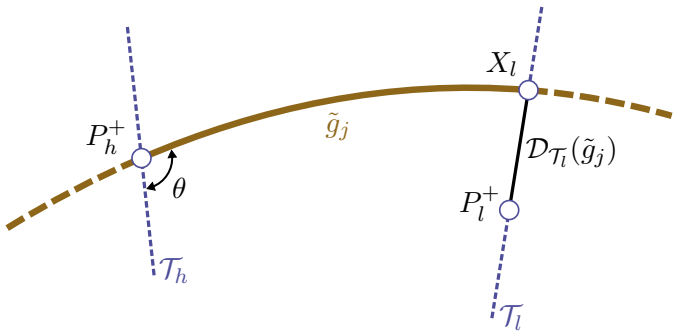


Figure 2.23: The signed distance function $\mathcal{D}_{\mathcal{T}_l}(\tilde{g}_j)$ is used for computing arc $\tilde{g}_j$ which is incident with P_h^+ and P_l^+.

P_h^+ of $g^+(s)$ on the neighboring trail curve $\mathcal{T}_h$. So $\tilde{g}_j$ is determined by the angle θ between $\tilde{g}_j$ and $\mathcal{T}_h$ (see Figure 2.22). The value of θ is obtained by minimizing target function (2.27).

3. There is more than one trail curve, and $\mathcal{T}_j$ is neither the first nor the last trail curve: $\tilde{g}_j$ needs to be incident with breakpoints P_h^+ and P_l^+ on neighboring trail curves $\mathcal{T}_h$ and $\mathcal{T}_l$. The geodesic curve containing $\tilde{g}_j$ is computed as a straightest geodesic traced from P_h^+ which minimizes a target function $e = [\mathcal{D}_{\mathcal{T}_l}(\tilde{g}_j)]^2$, where $\mathcal{D}_{\mathcal{T}_l}(\tilde{g}_j)$ is the signed distance along $\mathcal{T}_l$ from P_l^+ to the intersection point between $\mathcal{T}_l$ and $\tilde{g}_j$. The variable of this minimization problem is the intersection angle θ between $\mathcal{T}_l$ and $\tilde{g}_j$ (see Figure 2.23).

Resolving Intersections Between Trail Curves

A trail curve $\mathcal{T}_i$ may intersect a neighboring trail curve $\mathcal{T}_j$ at a point between the breakpoints P_i and P_i^+. This leads to undesired shape of $g^+(s)$ (see Figure 2.24(a)). For such intersections, we remove $\mathcal{T}_i$, $\mathcal{T}_j$, and insert a *virtual trail curve* $\overline{\mathcal{T}}$ which is incident with the center point $\overline{P}$ between P_i and P_j on $g(s)$. Then $g^+(s)$ is recomputed using the remaining trail curves as well as the virtual trail curve (see Figure 2.24(b)). This procedure is repeated until all intersections between neighboring trail curves are resolved.

Trimming Trail Curves

Two neighboring piecewise geodesic curves $g(s)$ and $g^+(s)$ can represent the boundary curves of a sequence of straight panels joined along end points. The trail curve segments between $g(s)$ and $g^+(s)$ represent the boundaries between neighboring panels. Therefore, we trim the trail curves by $g(s)$ and $g^+(s)$ to obtain these boundary curve segments. For an ordinary trail curve $\mathcal{T}_i$, we keep the part between breakpoints P_i and P_i^+. For a virtual trail curve $\overline{\mathcal{T}}$ which intersect $g^+(s)$ at breakpoint $\overline{P}^+$, we use geodesic curve segments to connect $\overline{P}^+$ with each break point on $g(s)$ associated with the trail curves that are

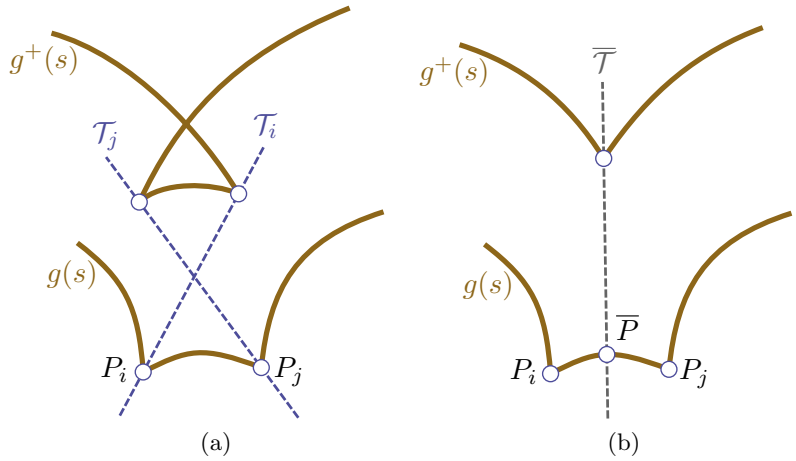


Figure 2.24: Resolving intersections of trail curves. (a) An intersection between neighboring trail curves $\mathcal{T}_i$ and $\mathcal{T}_j$, which leads to undesirable shapes of $g^+(s)$. (b) $\mathcal{T}_i$ and $\mathcal{T}_j$ are replaced by virtual trail curve $\overline{\mathcal{T}}$, and $g^+(s)$ is recomputed.

replaced by $\overline{\mathcal{T}}$ when resolving the trail curve intersections. Such connections result in triangle-shaped panels incident with the breakpoints on virtual curves (see Figure 2.25).

2.4.8 Results

Like geodesic curves, we can cover a surface patch by evolving piecewise geodesic curves from a source curve. In this section we give some examples of this type of evolution. In each example, the source curve $g^0(s)$ is a geodesic curve (which is also a piecewise

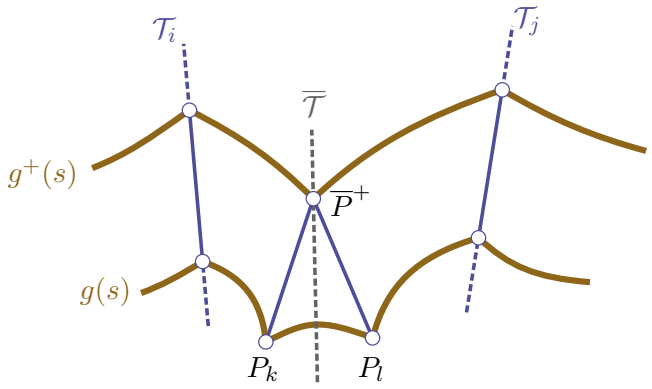


Figure 2.25: Trimming trail curves. The ordinary trail curves $\mathcal{T}_i$, $\mathcal{T}_j$ are simply trimmed by breakpoints. For virtual trail curve $\overline{\mathcal{T}}$, breakpoint $\overline{P}^+$ is connected to P_k and P_l, whose original trail curves have been replaced by $\overline{\mathcal{T}}$.

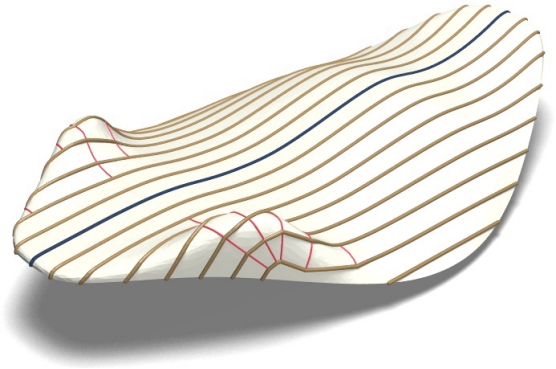


Figure 2.26: Piecewise geodesic curves (in brown) and trimmed trail curves (in red) evolved from a source curve (in dark blue). The evolution is performed on the same surface as Figure 2.14, using the same constant target distance function and the same source curve. The error threshold for introducing breakpoints is 5% of the target distance value.

geodesic curve with one arc), and the target distance function is a constant function. The computation of Jacobi fields requires specifying movement directions for the breakpoints. In all examples, we use the direction bisecting the angle between the two neighboring arcs as the movement direction.

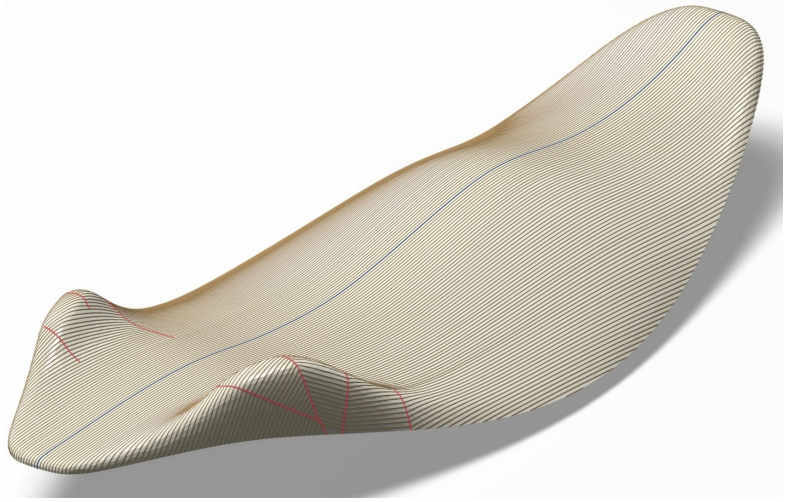
Figure 2.26 shows a layout of piecewise geodesic curves on the same surface as in Figure 2.14. The evolution starts from the same source curve, using the same target distance function. By evolving piecewise geodesic curves instead of geodesics, we get a more equidistant layout of curves, and get rid of intersections between neighboring curves.

Figure 2.27(a) shows the evolution of piecewise geodesic curves on the same surface as in Figure 2.26, using a smaller target distance value for the realistic width of wooden panels. Figures 2.27(b), 2.27(c) and 2.27(d) show a layout of wooden panels computed from these piecewise geodesic curves, using the *binormal method* in [Wallner et al. 2010].

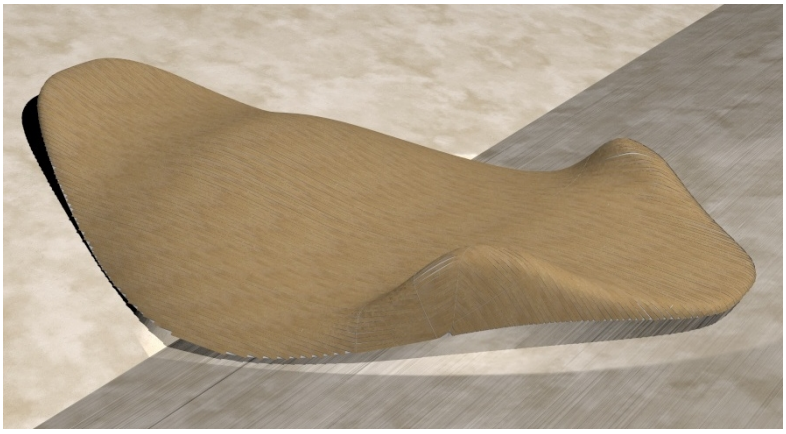
Figure 2.28 gives an example where the trail curves tend to converge and intersect. The intersections are resolved by the post-processing algorithm to produce nice layouts of piecewise geodesic curves and trimmed trail curves.

Figure 2.29 compares evolution of piecewise geodesic curves from the same source curve, using the same target distance value but different error thresholds for introducing breakpoints. The evolution with a smaller threshold produces a layout with smaller deviation from the target distance, and introduces more breakpoints.

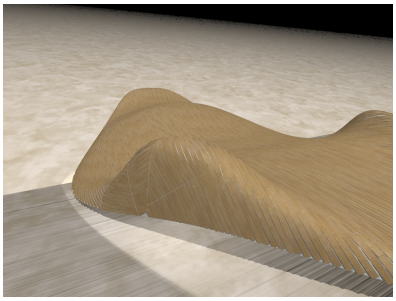
The timing data of some examples above is shown in Table 2.2. Here the meanings of #Face, δs, Time, %Jacobi,%NLS are the same as in Table 3.1. %Merge, %Int and %Trim are the percentages of time spent on merging neighboring arc, resolving intersection and trimming trail curves, respectively.



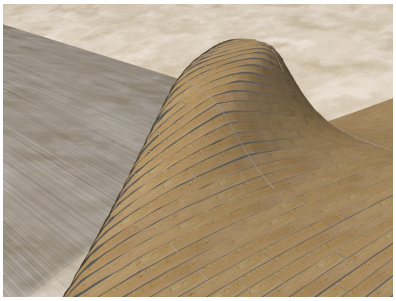
(a)



(b)

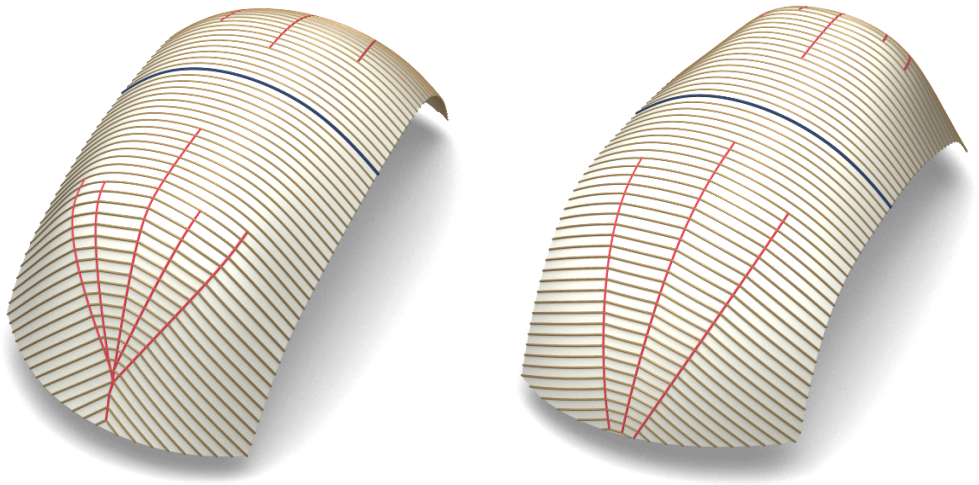


(c)



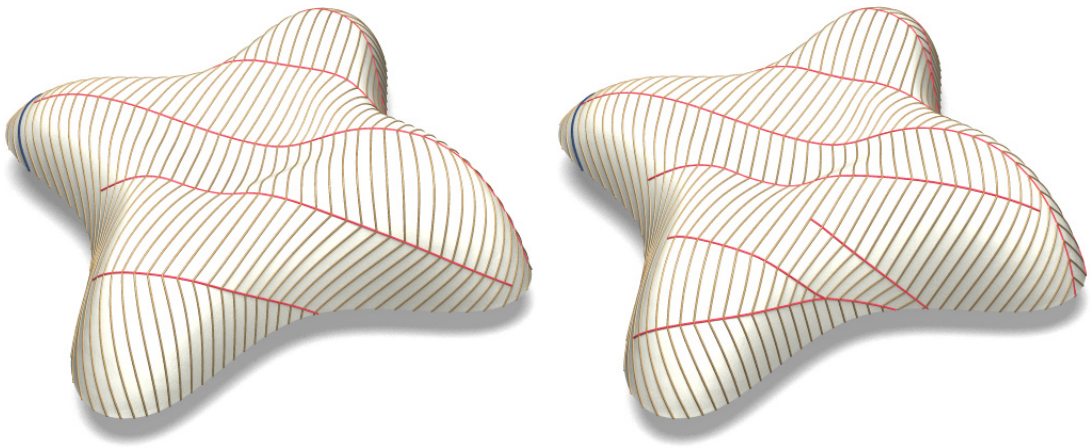
(d)

Figure 2.27: Wooden panel layout from piecewise geodesic curves. (a) A layout of piecewise geodesic curves by evolution. (b) The surface is covered by wooden panels of constant width, computed from the piecewise geodesic curves. (c)(d) Details of the wooden panels.



(a) (b)

Figure 2.28: A surface with intersecting trail curves, viewed from different angles. The error threshold for introducing breakpoints is 2.5% of the target distance value.



(a) (b)

Figure 2.29: Evolving piecewise geodesic curves on the same surface from the same source curve (blue, at extreme left), using the same target distance value and different error threshold values. The threshold values in (a) and (b) are 7.5% and 5% of the target distance value, respectively. The surface is taken from the top of the Lilium Tower, Warsaw, by Zaha Hadid Architects.

Fig	#Face	δs	Time	%Jacobi	%NLS	%Merge	%Int	%Trim
2.28	91180	$7.5 \cdot 10^{-4}$	48.2	0.33	47.75	36.34	2.74	0.15
2.29(a)	32018	$1.3 \cdot 10^{-3}$	31.0	0.45	64.44	26.52	0.61	0.16
2.29(b)	32018	$1.3 \cdot 10^{-3}$	29.8	0.44	64.76	25.66	1.37	0.13

Table 2.2: Timing data for evolution of piecewise geodesic curves.

2.5 Conclusion and Discussion

In this chapter, we present methods to evolve geodesics or piecewise geodesic curves on a surface patch, so that the distance function between neighboring curves are close to some specified target function. We employ the well-known Jacobi fields to approximate the distances between neighboring curves. Since the Jacobi fields on a geodesic form a two dimensional linear space, the problem of selecting Jacobi fields on geodesics or piecewise geodesic curves is reduced to solving a linear system. Using the selected Jacobi fields, a neighboring curve can be computed from the current curve by solving a non-linear least squares problem. Piecewise geodesic curves have more degrees of freedom than geodesics. So the evolution of piecewise geodesic curves provides more flexibility in creating geodesic patterns on a surface, producing layouts of curves which fit the target distance functions better. With the presented methods, we are able to create families of geodesics or piecewise geodesic curves with nearly constant distance between neighboring curves, as well as other interesting patterns of geodesics.

There are some limitations for these methods. First of all, Jacobi fields only provide first order approximations of distances between geodesic curves. There may be large approximation error, if the target distance is large, or if the underlying surface is not smooth. Besides, The current methods only work for surface patches without holes. Furthermore, these methods are essentially solving initial value problems, and the layout of the generated curves depends heavily on the source curve (see Figure 2.10 for an example of geodesics evolved from different source curves on the same surface). In order to find an optimal layout, a user may need to try different source curves and compare the results.

3 Functional Webs for Freeform Architecture

In this chapter, we investigate the problem of realizing architectural freeform surface shapes by regular patterns of long-range elements with special properties. Such long-range elements include structural elements and supports, floor levels, curved panels, and curves which are not physically realized by a single element but nevertheless are highly visible in the design. Since these elements can be represented as curves on the surface, we are investigating curve networks on a surface, where the curves have some special properties. This investigation is motivated by some recent architectural designs and constructions. In Figure 3.1, interwoven bamboo strips are used for an interior design. These strips can be modeled as geodesics on an underlying surface. Figure 3.2 shows a roof structure consisting of curve elements made from timber. Due to the general shapes of these elements, they have to be manufactured by CNC machining [Scheurer 2010]. In Figure 3.3, two families of wooden strips are imposed on a family of support elements, to form a timber rib shell [Pirazzi and Weinand 2006]. Here the wooden strips follow geodesics on the rib shell surface, and the support elements lie in vertical planes. Figure 3.4 is a tower with its outer shell built from laminated timber beams, where some joints of the beams are aligned with the horizontal floor levels. From these examples we see that certain properties of the curve elements can facilitate manufacturing, or provide desired structural properties. In this chapter we will consider the following special properties of the curves:

- *geodesic* curves are the shapes of straight panels subjected only to bending about their weak axis and to torsion [Pirazzi and Weinand 2006], which means that they can be manufactured from straight panels without CNC machining;
- *planar* curves are easily manufacturable for the simple reason that most factory floors are flat;
- *circular* curves, which are part of a circle, are even more efficiently manufacturable;
- *vertical* curves (contained in planes parallel to a hypothetical z axis) are useful for support elements;
- *horizontal* curves (contained in planes orthogonal to the hypothetical z axis) are useful for design and for floor levels to follow.

In accordance with the above examples, we consider three families of curves

$$\begin{aligned} \mathscr{F}_1 &= \{C_1(i)\}_{i\in\mathbb{Z}}, \\ \mathscr{F}_2 &= \{C_2(j)\}_{j\in\mathbb{Z}}, \\ \mathscr{F}_3 &= \{C_3(k)\}_{k\in\mathbb{Z}} \end{aligned} \tag{3.1}$$

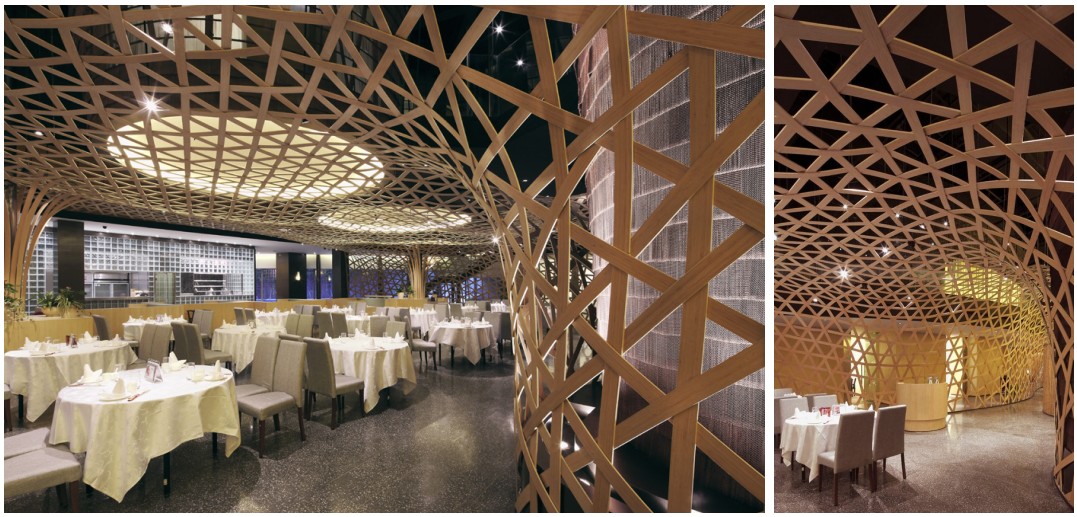


Figure 3.1: Interior design using composite bamboo strips, at Tang Palace, Hangzhou, China, designed by Atelier FCJZ.

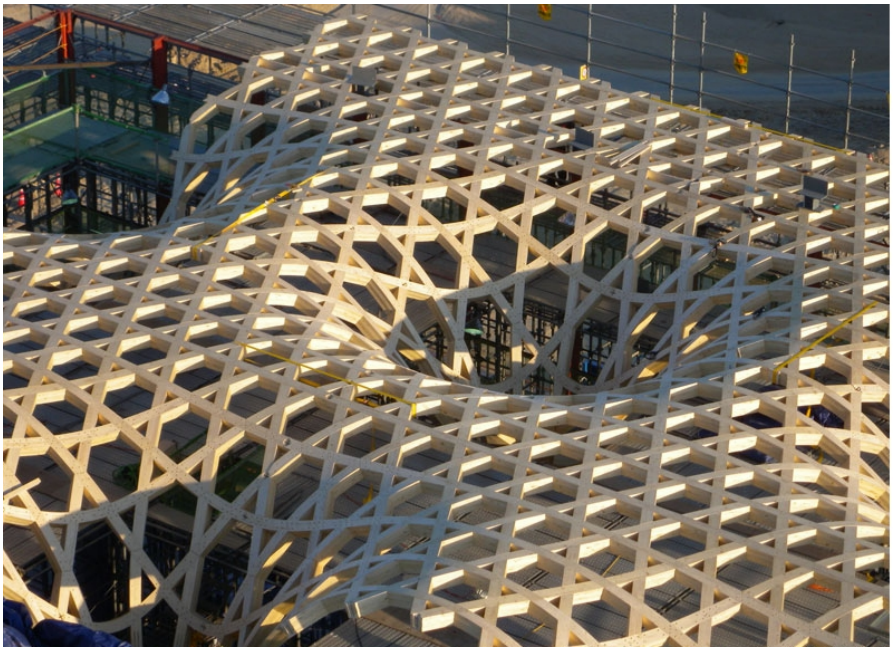
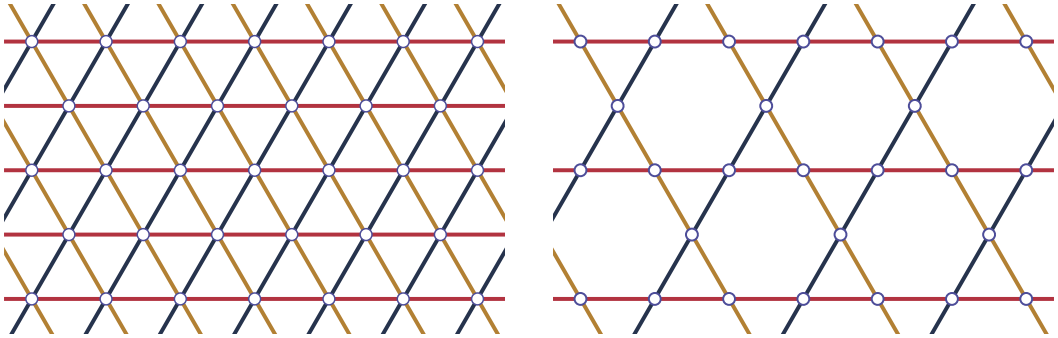


Figure 3.2: The roof structure of Yeoju Golf Club, South Korea, designed by Shigeru Ban, manufactured from timber using NC machining.

Figure 3.3: Assembling screw-laminated beams for a timber rib shell prototype based on two families of geodesics supported by a third family of vertical planar elements [Pirazzi and Weinand 2006]. Image courtesy IBOIS (EPFL, Lausanne).

Figure 3.4: The Korkeasaari Lookout Tower in Helsinki, Finland, with the outer shell built from laminated timber beams. Photos by Jussi Tiainen.



(a) (b)

Figure 3.5: Connectivity of the three families of curves $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$. (a) The curves are homeomorphic to three families of parallel lines, where any intersection point of two lines from two families is incident with a line from the remaining family. (b) Taking every second curve from each family, we obtain a tri-hex structure of curves as in Figure 3.2.

which lie on a surface, with regular connectivity. This means the following properties of the curves:

1. Two curves in the same family have no common point, i.e, $\mathcal{C}_\alpha(i) \cap \mathcal{C}_\alpha(j) = \varnothing$ $(\alpha = 1, 2, 3)$ if $i \neq j$;

2. For any intersection point between two curves from different families $p = \mathcal{C}_\alpha(i) \cap \mathcal{C}_\beta(j)$ $(\alpha \neq \beta)$, there is a unique curve $\mathcal{C}_\gamma(k)$ $(\gamma \neq \alpha, \beta)$ from the remaining family, such that $\mathcal{C}_\gamma(k)$ is incident with p;

3. For any intersection point p between two curves from different families, the tangents of the two curves at p are linearly independent.

$\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$ are homeomorphic to three families of parallel lines (see Figure 3.5(a)). Note that some special arrangements of curves can be obtained from the subsets of curves in $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$. For example, using every second curve in each family, we can obtain the tri-hex structure in Figure 3.2. (see Figure 3.5(b)).

In this chapter, we would like to compute the curve families $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$ such that the curves within each family have one of the special properties listed above. This task is converted into a numerical optimization problem, with appropriate target functionals expressing the desired properties for the curves. In terms of architectural design, we consider two types of problem:

1. In a *rationalization* problem, we are given a reference surface, and the computed discrete web needs to be close to this surface;

2. In a *form-finding* problem, only part of the web is required to satisfy some positional constraints (e.g., the boundary curves are required to be incident with some given shapes).

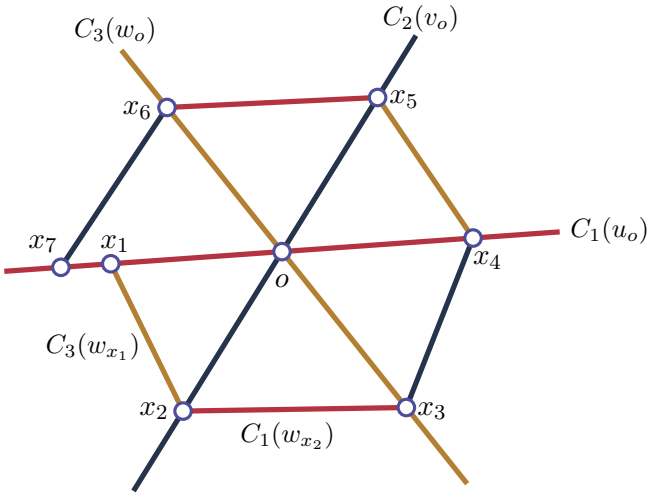


Figure 3.6: A hexagonal web satisfies the condition that for any point o and a neighboring point x_1, the point x_7 coincides with x_1.

Usually, rationalization problems are more restrictive, and may not be solvable.

3.1 Related Work

3.1.1 Web Geometry

It turns out that the curve families in Equation (3.1) can be seen as a discrete version of a *hexagonal web*, which was introduced by Blaschke and colleagues [Blaschke and Bol 1938] in a mathematical study which is now called *web geometry* [Chern 1982].

In web geometry, a *three-web* on an open domain D of a surface consists of three one-parameter families of smooth curves

$$\begin{aligned}\mathcal{C}_1 &= \{C_1(u)\}_{u\in\mathbb{R}},\\ \mathcal{C}_2 &= \{C_2(v)\}_{v\in\mathbb{R}},\\ \mathcal{C}_3 &= \{C_3(w)\}_{w\in\mathbb{R}}\end{aligned}$$

with the following properties [Pereira and Pirio 2009]:

1. $D = \cup_u C_1(u) = \cup_v C_2(v) = \cup_w C_3(w)$;
2. $C_i(s) \cap C_i(t) = \varnothing$ $(i = 1, 2, 3)$ if $s \neq t$;
3. For any intersection point p between two curves from different families, the tangents of the two curves at p are linearly independent.

The above properties imply that for any point $q \in D$, there exists exactly one curve from each family which is incident with q.

A *hexagonal web* is a three-web with the following closure property: Let o be a point in D, and $C_1(u_o)$, $C_2(v_o)$, $C_3(w_o)$ be the three curves in the web which are incident

with o. Take a point $x_1 \in C_1(u_o)$ in a neighborhood of o. According to the properties of three-webs, there exists a unique curve $C_3(w_{x_1}) \in \mathcal{C}_3$, which is incident with x_1 and intersects $C_2(v_o)$ at a point x_2. Similarly there exists a unique curve $C_1(u_{x_2}) \in \mathcal{C}_1$ which is incident with x_2 and intersects $C_3(w_0)$ at a point x_3. Repeating this construction, we get a sequence of points $x_1, \ldots, x_7$ which are incident with one of the three curves $C_1(u_o)$, $C_2(v_o)$, $C_3(w_o)$ (see Figure 3.6). If for any point $o \in D$ and any of its neighboring point $x_1 \in C_1(u_o)$, the above construction leads to a point $x_7 = x_1$, then this three-web is called a hexagonal web.

By comparing the definition of a three-web with the conditions we require for the curve families $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$ in Equation (3.1), we see that $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$ form a discrete three-web on a surface. Furthermore, it is trivial to verify that $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$ satisfy the closure property of hexagonal webs mentioned above. Therefore, we say that $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$ form a *discrete hexagonal web*. And the rationalization problem we consider here is directly analogous to the smooth problem of covering a given surface by a web whose curves enjoy certain properties. This is an old topic of differential geometry; unfortunately it has been solved only for some special cases. [Graf and Sauer 1924] prove that a complete hexagonal web of straight lines in $\mathbb{R}^2$ consists of the tangents of an algebraic curve of class 3 (possibly reducible). [Mayrhofer 1931] shows that surfaces of constant Gaussian curvature have the same variety of hexagonal webs of geodesics as the plane. For general surfaces, however, it is not known exactly which surfaces can be covered by a hexagonal web of geodesics: [Volk 1929] shows that for such surfaces, the coefficients of the first fundamental form must fulfill a certain third order PDE, for which a general solution looks hopeless. Likewise it is unknown which webs of planar curves a general surface can support (apart from the trivial cases of projecting webs of straight lines in $\mathbb{R}^2$ onto a surface). The very reduced problem of describing all webs in $\mathbb{R}^2$ formed by three linear pencils of circles has been solved only recently [Lazareva 1988] [Shelekhov 2007]. We conclude that many smooth problems of differential geometry which directly correspond to the problems studied here are unsolved, and it is difficult to obtain additional insight from the smooth case.

Geometry Processing

In the field of geometry processing, there is some work concerning curve networks with the same combinatorics as those considered here. [Wallner et al. 2007] consider such curve networks which minimize certain energy functionals. [Nieser et al. 2010] introduce hexagonal global parameterizations, where the parameter lines have the topology of hexagonal webs. Neither of them considers the functional properties which we aim at here. [Pottmann et al. 2010] construct a hexagonal web of approximately geodesic curves on a mesh surface by computing three scalar functions on the surface, with the level set curves of each function being approximately geodesic. For these level set curves to form a hexagonal web, the function values need to satisfy a certain algebraic condition. The three scalar functions are determined by minimizing a target functional which penalizes the deviation of the level set curves from geodesics. In this chapter, we compute geodesic webs in a different way, i.e., by searching for regular triangle meshes whose edge polylines

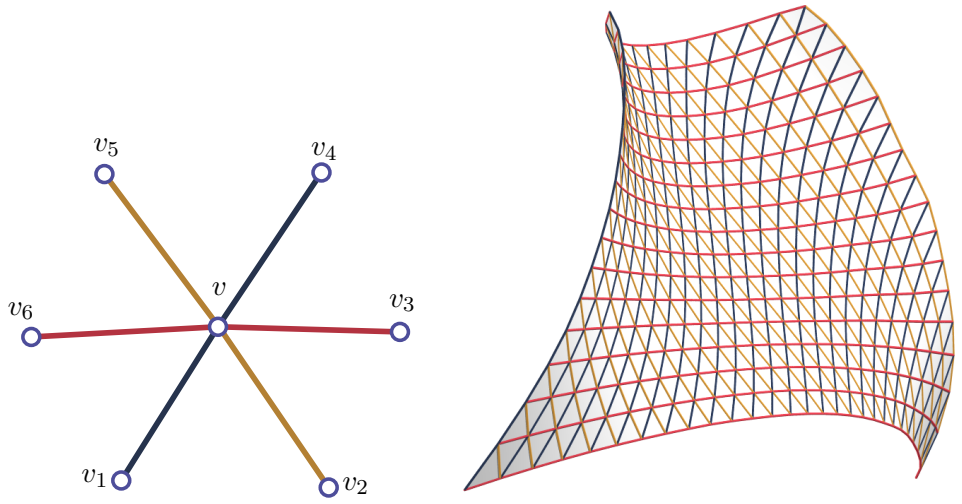


Figure 3.7: A discrete hexagonal web can be represented by a regular triangle mesh. (a) For an interior vertex v, neighboring vertices v_i and v_{i+3} $(i = 1, 2, 3)$ belong to the same polyline which represents a curve in the web. (b) A simple regular mesh which represents a web. The blue, yellow and red families of polylines represent planar, geodesic and circular curves, respectively.

are straightest geodesic curves on the mesh surface. Such an approach enables us to consider also hexagonal webs with other types of curves, such as planar curves and circular curves, and provides a general framework for computing discrete webs.

3.2 Discrete Webs by Global Optimization

In this section we present a method to construct discrete hexagonal webs by optimizing a target functional. The key idea is that a discrete hexagonal web can be discretized as a regular triangle mesh (i.e., each interior vertex is of valence 6), such that the edge polylines of the mesh represent the curves of the web, while the mesh vertices represent the intersection points between the curves. Then the underlying surface of the web is represented by the mesh surface. Among the six neighboring vertices v_i $(i = 1, \ldots, 6)$ of an interior vertex v, two opposite ones v_i and v_{i+3} $(i = 1, 2, 3)$ belong to the same edge polyline representing a curve (see Figure 3.7(a)). Using this rule, we can infer three families of edge polylines $\mathscr{L}_1$, $\mathscr{L}_2$, $\mathscr{L}_3$ corresponding to the three families of curves $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$ in Equation (3.1) (see Figure 3.7(b)). Therefore, the problem of computing a web with specified properties of curves is converted to a problem of computing a regular triangle mesh whose edge polylines have the specified properties. In a rationalization problem, the mesh needs to approximate the given surface. In a form-finding problem, the mesh needs to satisfy the given positional constraints. Such a mesh is determined

by minimizing a target functional

$$f = \sum_j \lambda_{\text{prop}_j} f_{\text{prop}_j}(\mathscr{L}_j) + \sum_j \lambda_{\text{fair}_j} f_{\text{fair}_j}(\mathscr{L}_j) + \lambda_{\text{prox}} f_{\text{prox}} + \lambda_{\text{bdry}} f_{\text{bdry}}. \tag{3.2}$$

Here $f_{\text{prop}_j}(\mathscr{L}_j)$ refers to a functional which penalizes the deviation of family $\mathscr{L}_j$ from its required property. $f_{\text{fair}_j}(\mathscr{L}_j)$ is a fairness functional for family $\mathscr{L}_j$. f_{prox} penalizes the deviation between the mesh surface and the constraint shapes which the mesh surface needs to be close to. f_{bdry} penalizes the deviation between the mesh boundary curves and the target boundary curves. f_{prox} and f_{bdry} express the positional constraints imposed on the web surface by a rationalization or form-finding problem. The λ's represent weights. The variables for this minimization problem are the mesh vertex coordinates, as well as the auxiliary variables required by the functionals. This minimization problem is solved using numerical optimization. The formulation of each functional is explained below.

3.2.1 The Fairing Functional

For a polyline L, we employ second order or third order differences of consecutive vertices in L as a fairing functional

$$\begin{aligned} h_{\text{fair,II}}(L) &= \sum_{v_1,v_2,v_3 \text{ consecutive in } L} \|v_1 - 2v_2 + v_3\|^2, \\ h_{\text{fair,III}}(L) &= \sum_{v_1,v_2,v_3,v_4 \text{ consecutive in } L} \|v_1 - 3v_2 + 3v_3 - v_4\|^2. \end{aligned} \tag{3.3}$$

The fairing functional for a polyline family $\mathscr{L}$ is defined as the sum of one of the above terms in (3.3) for all polylines in $\mathscr{L}$

$$f_{\text{fair},\sigma}(\mathscr{L}) = \sum_{L \in \mathscr{L}} h_{\text{fair},\sigma}(L), \quad \sigma = \text{II}, \text{III}. \tag{3.4}$$

We may choose different σ in (3.4) for different families of polylines. Functional $h_{\text{fair,II}}(L)$ reaches a minimum when the vertices of L are uniformly distributed on a straight line, while $h_{\text{fair,III}}(L)$ is minimized only if the vertices of L lie on a parabola or a straight line. $h_{\text{fair,II}}(L)$ is more restrictive since it tends to "straighten" the polyline. On the other hand, $h_{\text{fair,II}}(L)$ is able to produce more evenly spaced vertices on the polyline. The choice between $h_{\text{fair,II}}(L)$ and $h_{\text{fair,III}}(L)$ depends on the specific requirement from the actual problem.

3.2.2 The Shape Proximity Functional

Functional f_{prox} expresses the deviation between the web surface mesh and the constraint shapes. For a rationalization problem, the constraint shape is a given surface which needs to be approximated by the web. For a form-finding problem, the constraint shapes can be some curves or points with which the web surface needs to be incident. f_{prox} can have one of the following two forms:

The Proximity Functional using Sample Points of the Constraint Shapes

The first form of f_{prox} is defined using the shortest distance from a set of sample points $\mathscr{P}$ of the constraint shapes to the mesh M representing the web

$$f_{\text{prox,I}} = \sum_{p \in \mathscr{P}} [\text{dist}(p, M)]^2. \tag{3.5}$$

Here $\text{dist}(p, M)$ is the shortest distance from point p to M. It is evaluated by searching for a *footpoint* $p_\perp$ on M, whose distance to p is the smallest among all points of M. Then $\text{dist}(p, M) = \|p - p_\perp\|$. To minimize the target functional (3.2) by numerical optimization, we also need to compute the derivatives of $\text{dist}(p, M)$ with respect to the variables of the minimization problem. When the variable values undergo small enough changes, we can obtain closed-form representations of $\text{dist}(p, M)$, from which we can compute the derivatives with respect to the current variable values. Depending on the footpoint position $p_\perp^0$ on the current mesh, there are three cases to consider:

1. $p_\perp^0$ is in the interior of a face F of mesh M: For small enough changes of vertex positions of M, the footpoint of p on the new mesh will still lie on F. Therefore, locally $\text{dist}(p, M)$ equals the distance from p to the plane containing F. Let v_1, v_2, v_3 be the vertices of F, then $\text{dist}(p, M)$ has a local representation

$$\text{dist}(p, M) = |(p - v_1) \cdot N_F|, \tag{3.6}$$

   where

$$N_F = \frac{(v_2 - v_1) \times (v_3 - v_1)}{\|(v_2 - v_1) \times (v_3 - v_1)\|},$$

   is the unit normal vector of the face. Note that here $\text{dist}(p, M)$ only depends on the coordinates of v_1, v_2, v_3. The derivative of $\text{dist}(p, M)$ with respect to any other variable is zero.

2. $p_\perp^0$ is in the interior of an edge E of M: For small enough changes of vertex positions of M, the new footpoint still lies on E, unless p is on a plane which is incident with E and orthogonal to a face containing E. Since such a case happens with probability zero, it can be ignored in practice. Therefore, $\text{dist}(p, M)$ locally equals the distance from p to the straight line containing E. Let v_1, v_2 be the two vertices of E, then

$$\text{dist}(p, M) = \|(p - v_1) - T_E(T_E \cdot (p - v_1))\|,$$

   where

$$T_E = \frac{v_2 - v_1}{\|v_2 - v_1\|}$$

   is a unit tangent vector of the edge. Here $\text{dist}(p, M)$ only depends on the coordinates of v_1 and v_2.

3. $p_\perp^0$ coincides with a vertex v of M: For small enough changes of vertex positions of M, the new footpoint still coincides with v, unless p is on the surface of the following pyramid $\widehat{P}$: the apex of $\widehat{P}$ is at v; the base polygon of $\widehat{P}$ is an n_v-gon at infinity, where n_v is the valence of v; there is a one-to-one correspondence between the pyramid faces $\{F_i\}$ incident with the apex, and the mesh edges $\{E_i\}$ incident with v, so that E_i is orthogonal to F_i. Point p lies on the surface of $\widehat{P}$ with probability zero. Therefore, $\mathrm{dist}(p, M)$ locally equals the distance from p to v

$$\mathrm{dist}(p, M) = \|p - v\|.$$

This function only depends on the coordinates of v.

The Proximity Functional using Vertices of the Web Surface

The second form of f_{prox} can be used in rationalization problems, where the constraint shape is a given surface S. It is defined using the tangential distance from the vertices $\{v\}$ of the web surface mesh M to S

$$f_{\mathrm{prox,II}} = \sum_{v \in M} [\mathrm{dist}_T(v, S)]^2. \tag{3.7}$$

Here the tangential distance $\mathrm{dist}_T(v, S)$ is defined as the signed distance from vertex v to the tangent plane of S at $v_\perp$, where $v_\perp$ is the footpoint of v on S. $\mathrm{dist}_T(v, S)$ only depends on the coordinates of v. To compute the derivatives of $\mathrm{dist}_T(v, S)$, we ignore the dependence of the footpoint $v_\perp$ on the position of v, and $\mathrm{dist}_T(v, S)$ becomes a linear function of the coordinates of v. The square of this tangential distance is suggested by [Pottmann and Leopoldseder 2003] as an approximation of the true squared distance function (SQDF) from v to S, if the footpoint $v_\perp$ is not on the boundary of S. It is in second order agreement with the SQDF if v is close to S [Pottmann and Leopoldseder 2003].

Comparison Between the Two Forms

$f_{\mathrm{prox,II}}$ can only be used for rationalization problems, while $f_{\mathrm{prox,I}}$ can be used in rationalization as well as form-finding. Besides, $f_{\mathrm{prox,I}}$ requires the web surface to cover the whole constraint shapes: If we sample the constraint shapes densely enough, minimizing functional $f_{\mathrm{prox,I}}$ requires every sample point to be close to the web surface mesh, effectively covering the constraint shapes with the web surface. Such a covering effect can not be guaranteed by $f_{\mathrm{prox,II}}$: For a given constraint surface S, a web surface mesh M with all vertices lying on a sub-region of S leads to a minimum of $f_{\mathrm{prox,II}}$. However, S is not entirely covered by M.

3.2.3 The Boundary Proximity Functional

For a boundary polyline B of the web surface mesh, and a curve $\mathcal{C}_B$ as the target position of B, the following functional penalizes the deviation between B and $\mathcal{C}_B$

$$h_{\text{bdry}}(B) = \sum_{v \in B} [\text{dist}_T(v, \mathcal{C}_B)]^2. \tag{3.8}$$

Here v is a vertex on polyline B, $\text{dist}_T(v, \mathcal{C}_B)$ is the distance from v to the tangent line of $\mathcal{C}_B$ at the footpoint of v. The value and derivatives of $\text{dist}_T(v, \mathcal{C}_B)$ is evaluated in a way similar to $\text{dist}_T(v, S)$ in (3.7). The boundary proximity functional f_{bdry} is the sum of the functional (3.8) for all boundary curves of the web surface

$$f_{\text{bdry}} = \sum_{B} h_{\text{bdry}}(B).$$

3.2.4 The Planar Property

For a polyline to be a planar curve, all of its vertices need to lie on the same plane. A plane in $\mathbb{R}^3$ can be represented by a unit normal vector n and a scalar d, such that any point p on the plane satisfies

$$p \cdot n - d = 0. \tag{3.9}$$

Since $p \cdot n - d$ is the signed distance from point p to the plane, we use the following functional to penalize the deviation of a polyline from a plane

$$h_{\text{planar}}(L) = \sum_{v \in L} (n_L \cdot v - d_L)^2. \tag{3.10}$$

Here unit vector n_L and scalar d_L are auxiliary variables representing the plane. The planar property functional for a polyline family $\mathscr{L}$ is defined as

$$f_{\text{planar}}(\mathscr{L}) = \sum_{L \in \mathcal{I}_{\text{planar}}(\mathscr{L})} f_{\text{planar}}(L).$$

Here $\mathcal{I}_{\text{planar}}(\mathscr{L})$ is the set of polylines in $\mathscr{L}$ which need to be planar and have more than three vertices. We only consider polylines with more than three vertices, because a polyline with less than four vertices is always planar. For numerical optimization, the initial values of n_L and d_L in (3.10) are obtained from the best fitting plane to the initial vertex positions in L, using principal component analysis (PCA) [Jolliffe 2002].

Special Positions of Planes

For a polyline L to lie in a horizontal plane, we note that a horizontal plane has constant unit normal vector $n = (0, 0, 1)$, and functional (3.10) is modified to

$$h^H_{\text{planar}}(L) = \sum_{v \in L} (v_z - d_L)^2,$$

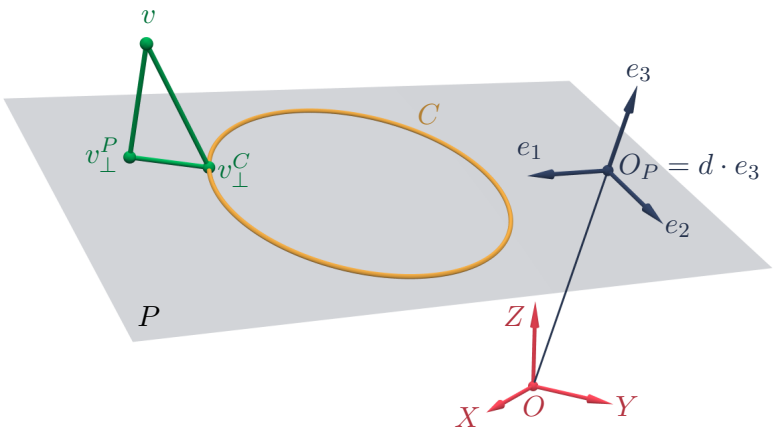


Figure 3.8: Distance from a point v to a circle C. The plane P which contains the circle is determined from an orthonormal frame e_1, e_2, e_3 at point $O_P = d \cdot e_3$. Plane P is incident with O_P and has normal vector e_3. The distance from v to its footpoint $v_\perp^C$ on C can be decomposed as $\|v - v_\perp^C\|^2 = \|v - v_\perp^P\|^2 + \|v_\perp^P - v_\perp^C\|^2$, where $v_\perp^P$ is the footpoint of v on P.

where v_z is the z coordinate of vertex v, and d_L is the only auxiliary variable. For L to be in a vertical plane, we note that a vertical plane has unit normal vector $n = (n_x, n_y, 0)$, and functional (3.10) is modified to

$$h^V_{\text{planar}}(L) = \sum_{v \in L} (n_{L,x} v_x + n_{L,y} v_y - d_L)^2,$$

where v_x, v_y are the x and y coordinates of vertex v, and scalars $n_{L,x}$, $n_{L,y}$, d_L are auxiliary variables with the constraint $(n_{L,x})^2 + (n_{L,y})^2 = 1$. Also note that although a polyline with no more than three vertices is always planar, it may not be in a vertical plane or horizontal plane. Therefore, for functionals $h^H_{\text{planar}}(L)$ and $h^V_{\text{planar}}(L)$, we cannot ignore polylines with less than four vertices.

3.2.5 The Circular Property

For a polyline L to represent a circular curve, we require all the vertices of the polyline to be on a circle. Therefore, the following functional penalizes the deviation of a polyline L from a given circle C

$$h_{\text{circle}}(L) = \sum_{v \in L} [\text{dist}(v, C)]^2, \tag{3.11}$$

where $\text{dist}(v, C)$ is the distance from a vertex v to its footpoint $v_\perp^C$ on circle C. Note that the squared distance $[\text{dist}(v, C)]^2$ can be decomposed into two parts (see Figure 3.8)

$$[\text{dist}(v, C)]^2 = \left\|v - v_\perp^C\right\|^2 = \left\|v - v_\perp^P\right\|^2 + \left\|v_\perp^P - v_\perp^C\right\|^2 = [\text{dist}(v, P)]^2 + [\text{dist}(v_\perp^P, C)]^2.$$

Here $v_\perp^P$ is the footpoint of v on the plane P which contains circle C, $\text{dist}(v, P) = \left\|v - v_\perp^P\right\|$ is the distance from v to the plane P, and $\text{dist}(v_\perp^P, C) = \left\|v_\perp^P - v_\perp^C\right\|$ is the

distance from $v_\perp^P$ to circle C. Plane P can be represented by a unit normal vector e_3 and a scalar d, so that

$$[\mathrm{dist}(v, P)]^2 = (v \cdot e_3 - d)^2.$$

Furthermore, a pair of orthonormal vectors e_1, e_2 parallel with P leads to a local coordinate system (ξ_1, ξ_2) in P, with the origin at $O_P = de_3$, and the two coordinate axes parallel to e_1 and e_2, respectively. In this local coordinate system, the footpoint $v_\perp^P$ has coordinates

$$\xi_1(v_\perp^P) = v \cdot e_1, \quad \xi_2(v_\perp^P) = v \cdot e_2. \tag{3.12}$$

Besides, in this coordinate system, the circle C can be represented by an equation

$$g(\xi_1, \xi_2) = b_0(\xi_1^2 + \xi_2^2) + b_1\xi_1 + b_2\xi_2 + b_3 = 0, \tag{3.13}$$

where scalars b_i $(i = 0, 1, 2, 3)$ satisfy a normalization constraint

$$b_1^2 + b_2^2 - 4b_0b_3 = 1. \tag{3.14}$$

Using this representation, the squared distance $[\mathrm{dist}(v_\perp^P, C)]^2$ can be approximated by $[g(\xi_1(v_\perp^P), \xi_2(v_\perp^P))]^2$ [Pratt 1987]. Here we do not use the center position and the radius value to represent a circle, because we want to include straight lines as special cases of circles, which cannot be represented using the center and the radius. Now functional (3.11) can be written as

$$h_{\mathrm{circle}}(L) = \sum_{v \in L} (v \cdot e_3 - d)^2 + [g(\xi_1(v_\perp^P), \xi_2(v_\perp^P))]^2. \tag{3.15}$$

Note that e_1, e_2, e_3 form an orthonormal frame, which can be encoded by a unit quaternion [Kuipers 2002]. Therefore, the auxiliary variables for (3.15) are scalars d, b_i $(i = 0, 1, 2, 3)$, and the unit quaternion representing e_1, e_2, e_3. For initial values of the auxiliary variables, we first compute a best fitting plane for the vertices of L using PCA, which gives e_3 and d. Any pair of orthonormal vectors in this plane can be chosen as e_1 and e_2, giving the initial value of the unit quaternion. Then we compute the local coordinates for the footpoints of the vertices of L on plane P, according to (3.12). The initial values of b_i $(i = 0, 1, 2, 3)$ are computed by fitting a circle to these footpoints using a metric derived from (3.13) [Chernov 2010]. Using (3.15), the circular property functional for a polyline family $\mathscr{L}$ is defined as

$$f_{\mathrm{circle}}(\mathscr{L}) = \sum_{L \in \mathcal{I}_{\mathrm{circle}}(\mathscr{L})} h_{\mathrm{circle}}(L),$$

where $\mathcal{I}_{\mathrm{circle}}(\mathscr{L})$ are the set of polylines in $\mathscr{L}$ which are required to be circular and have more than three vertices. We ignore polylines with no more than three vertices because their vertices are always on the same circle.

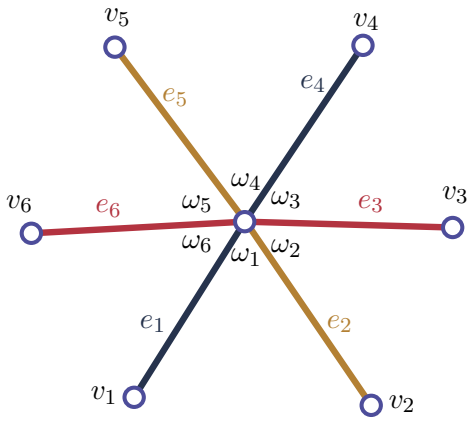


Figure 3.9: The geodesic condition of a mesh polyline at a vertex is defined using the angles $\{\omega_i\}_{i=1,\dots,6}$ formed by the neighboring edges $\{e_i\}_{i=1,\dots,6}$ of the vertex.

3.2.6 The Geodesic Property

When dealing with geodesic mesh polylines we employ the concept of straightest geodesics introduced by [Polthier and Schmies 1998]. For an interior vertex v of a regular mesh, there are six edges $e_i = \overline{vv_i} (i = 1, \dots, 6)$ incident with v (see Figure 3.9). Each pair of opposite edges e_i and e_{i+3} (indices modulo 6) belongs to a polyline from one of the three families. Let $\omega_i \in [0, \pi)$ be the angle between edges e_i and e_{i+1}. Then the polyline which contains e_i and e_{i+3} is a straightest geodesic at v if [Polthier and Schmies 1998]

$$\omega_i + \omega_{i+1} + \omega_{i+2} = \omega_{i+3} + \omega_{i+4} + \omega_{i+5}. \tag{3.16}$$

Accordingly the following functional expresses the straightest geodesic condition at v for the polyline containing e_i and e_{i+3}

$$h_{\text{geod}}(v, i) = (\sum_{j=0}^{2} \omega_{i+j} - \sum_{k=3}^{5} \omega_{i+k})^2. \tag{3.17}$$

Note that since

$$\omega_i = \arccos\left(\frac{v_i - v}{\|v_i - v\|} \cdot \frac{v_{i+1} - v}{\|v_{i+1} - v\|}\right),$$

functional $h_{\text{geod}}(v, i)$ depends on the positions of v and its one-ring neighbor vertices. For a polyline L to be a straightest geodesic, condition (3.16) needs to be satisfied at every interior mesh vertex of L. The following functional penalizes the deviation between L and a straightest geodesic

$$h_{\text{geod}}(L) = \sum_{v \in \mathcal{V}(L)} h_{\text{geod}}(v, i_v(L)), \tag{3.18}$$

where $\mathcal{V}(L)$ is the set of interior mesh vertices on L, and $i_v(L)$ is the smallest index of a neighboring edge of v which belongs to polyline L. The geodesic property functional

for a polyline family $\mathscr{L}$ is defined as

$$f_{\text{geod}}(\mathscr{L}) = \sum_{L \in \mathcal{I}_{\text{geod}}(\mathscr{L})} h_{\text{geod}}(L),$$

where $\mathcal{I}_{\text{geod}}(\mathscr{L})$ is the set of polylines in $\mathscr{L}$ which are required to be geodesic.

3.2.7 Counting Degrees of Freedom

For a mesh with N_V vertices, there are $N_f = 3N_V$ degrees of freedom (d.o.f) for vertex coordinates. The planar, circular and geodesic properties of mesh edge polylines induce constraints on the mesh vertex coordinates. If the effective number of these constraints is larger than N_f, the minimization of functional (3.2) may not produce satisfactory results. Here we count the number of these constraints, to see what properties we can require for a web.

The Planar Property

For a polyline L which is required to be planar, each vertex needs to satisfy a condition (3.9). The effective number of constraints for L is

$$C_{\text{planar}}(L) = N_V^L - N_{\text{aux}}^L,$$

where N_V^L is the number of vertices on L, and N_{aux}^L is the number of d.o.f. provided by the auxiliary variables for the plane. For a general plane, $N_{\text{aux}}^L = 3$; for a vertical plane, $N_{\text{aux}}^L = 2$; for a horizontal plane, $N_{\text{aux}}^L = 1$. For all polylines of a family $\mathscr{L}$ to be planar, the total number of constraints is

$$C_{\text{planar}}(\mathscr{L}) \quad = \quad \sum_{\substack{L \in \mathscr{L}, \\ N_V^L \geq m_L}} C_{\text{planar}}(L) \quad \leq \quad N_V - \sum_{\substack{L \in \mathscr{L}, \\ N_V^L \geq m_L}} N_{\text{aux}}^L.$$

Here m_L is the required minimum number of vertices for L. $m_L = 2$ if L is contained in a vertical or horizontal plane, and $m_L = 4$ if L is in a general plane.

A web where all curves are planar is called a *planar web*. For a regular mesh to represent a planar web, all edge polylines need to be planar, and the number of constraints induced by the planar property is

$$\sum_{i=1,2,3} C_{\text{planar}}(\mathscr{L}_i) \quad \leq \quad 3N_V - \sum_{N_V^L > m_L} N_{\text{aux}}^L.$$

The difference between the number of d.o.f. from vertex coordinates and the number of constraints from the planar property is

$$N_f - \sum_{i=1,2,3} C_{\text{planar}}(\mathscr{L}_i) \quad \geq \quad \sum_{N_V^L > m_L} N_{\text{aux}}^L.$$

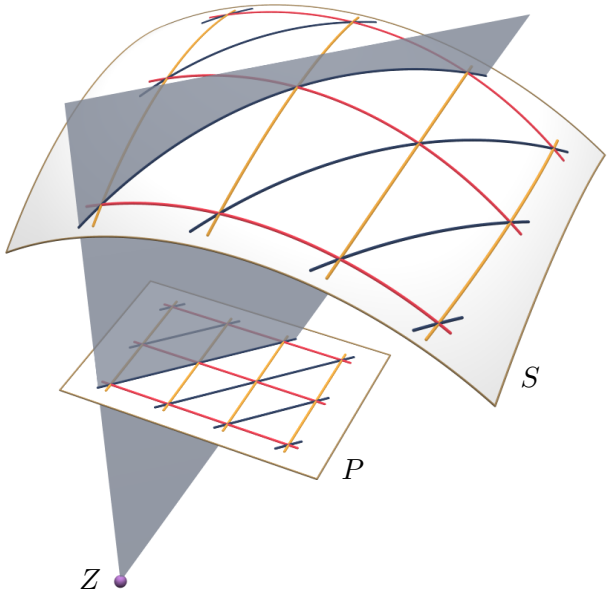


Figure 3.10: A trivial planar web. A web of straight lines in plane P is projected to a surface S, using point Z as the center of projection. This results in a trivial planar web on S.

Therefore, we still have some degrees of freedom for the mesh to satisfy the positional constraints imposed by a rationalization or form-finding problem. Indeed, every surface S can be covered by a planar web in a trivial way: Take a plane P with a web $\mathcal{W}$ of straight lines on it, and a point Z which is not incident with plane P. Each straight line L in the web $\mathcal{W}$, together with point Z, determine a plane P_L. The intersection between P_L and the surface S is a planar curve on S, which we call the projection of L from center Z onto S. The projections of all straight lines in $\mathcal{W}$ from Z onto S is a planar web on S (see Figure 3.10). We call such a planar web *trivial*. A planar web is trivial if there exists a point Z such that for every curve C in the web, there is a plane which contains C and Z. In some cases we can even rationalize a surface with a planar web which is not trivial. See Figure 3.15 for an example.

The Circular Property

The circular property of a polyline L requires each vertex of L to satisfy two conditions: (1) the vertex lies on the plane of the circle; (2) the footpoint of the vertex in this plane satisfies Equation (3.13). The number of d.o.f. provided by the auxiliary variables is six. Therefore, the effective number of constraints for L is

$$C_{\text{circle}}(L) = 2N_V^L - 6.$$

If we want to construct a mesh where all polylines are circular, the total number of constraints is

$$C_{\text{circle}} \quad = \quad \sum_{N_V^L>3} (2\mathcal{N}_V(L) - 6).$$

This number is often larger than N_f. So the circular property is more restrictive than the planar property, and we can construct webs of circular curves only in some special cases.

The Geodesic Property

The geodesic property of a polyline L induces one constraint (3.16) for each interior mesh vertex of L. Therefore, for all members of a polyline family $\mathscr{L}$ to be geodesic, the number of constraints $C_{\text{geod}}(\mathscr{L})$ equals the number of interior vertices N_{IV} of the mesh. For a geodesic web (i.e., all polylines are geodesic), the total number of constraints is

$$\sum_{i=1,2,3} C_{\text{geod}}(\mathscr{L}_i) = 3N_{IV}.$$

The difference between N_f and the total number of constraints is

$$N_f - \sum_{i=1,2,3} C_{\text{geod}}(\mathscr{L}_i) = 3N_V - 3N_{IV} = 3N_{BV},$$

where N_{BV} is the number of boundary vertices of the mesh. In other words, we have as many degrees of freedom as the coordinates of boundary vertices. We can therefore expect that *a geodesic web is uniquely determined by its boundary.* It is possible to construct a geodesic web which approximates a given surface (see Figure 3.11 for an example).

3.2.8 Implementation Details

Numerical Minimization of the Target Functional

The minimization of functional (3.2) is a non-linear least squares problem. We solve it using the Levenberg-Marquardt (LM) method [Madsen et al. 2004]. The first order derivative values required by the LM method are computed analytically. Since each squared component of the target functional only depends on a small number of variables, we need to solve a sparse linear system in each iteration. This is done using the sparse Cholesky factorization in [Chen et al. 2008].

Handling Auxiliary Variables with Normalization Constraints

Some auxiliary variables are subject to normalization constraints. For example, the coordinate variables $n_{L,x}, n_{L,y}, n_{L,z}$ of the unit normal vector n_L in (3.10) needs to satisfy $(n_{L,x})^2 + (n_{L,y})^2 + (n_{L,z})^2 = 1$. This side condition complicates the computation

of derivatives. To make $n_{L,x}, n_{L,y}, n_{L,z}$ independent of each other, we represent the unit vector n_L as

$$n_L = \frac{(n_{L,x}, n_{L,y}, n_{L,z})}{\sqrt{(n_{L,x})^2 + (n_{L,y})^2 + (n_{L,z})^2}}.$$

This representation incorporates the normalization condition. The derivatives of n_L with respect to $n_{L,x}, n_{L,y}, n_{L,z}$ can be computed analytically, as long as $(n_{L,x})^2 + (n_{L,y})^2 + (n_{L,z})^2 > 0$. At the start of the LM method, the initial values $n_{L,x}^{(0)}, n_{L,y}^{(0)}, n_{L,z}^{(0)}$ are normalized. In each iteration of the LM method, a step ΔX is computed for all variables. Let $\delta n_{L,x}, \delta n_{L,y}, \delta n_{L,z}$ be the be the components of ΔX for $n_{L,x}, n_{L,y}, n_{L,z}$, respectively, and let $n_{L,x}^{(k)}, n_{L,y}^{(k)}, n_{L,z}^{(k)}$ be the corresponding current values of variables. We compute the candidate values

$$\begin{aligned} n_{L,x}^{(\text{new})} &= n_{L,x}^{(k)} + \delta n_{L,x}, \\ n_{L,y}^{(\text{new})} &= n_{L,y}^{(k)} + \delta n_{L,y}, \\ n_{L,z}^{(\text{new})} &= n_{L,z}^{(k)} + \delta n_{L,z}, \end{aligned}$$

and check for the condition

$$\left(n_{L,x}^{(\text{new})}\right)^2 + \left(n_{L,y}^{(\text{new})}\right)^2 + \left(n_{L,z}^{(\text{new})}\right)^2 > \varepsilon \tag{3.19}$$

where $\varepsilon > 0$ is a small threshold. We modify the step control algorithm for the LM method, so that if condition (3.19) is violated, the step ΔX is rejected and a new step is computed. If a step is accepted, $n_{L,x}^{(\text{new})}$, $n_{L,y}^{(\text{new})}$, $n_{L,z}^{(\text{new})}$ are normalized to become the values of $n_{L,x}, n_{L,y}, n_{L,z}$ in the next iteration

$$n_{L,x}^{(k+1)} = \frac{n_{L,x}^{(\text{new})}}{D}, \quad n_{L,y}^{(k+1)} = \frac{n_{L,y}^{(\text{new})}}{D}, \quad n_{L,z}^{(k+1)} = \frac{n_{L,z}^{(\text{new})}}{D},$$

where $D = \sqrt{\left(n_{L,x}^{(\text{new})}\right)^2 + \left(n_{L,y}^{(\text{new})}\right)^2 + \left(n_{L,z}^{(\text{new})}\right)^2}$. All variables that are subject to normalization constraints are handled in this way.

Evaluation of Footpoints

Some functionals require searching for a footpoint $p_\perp$ of given point p on a surface or a curve. We assume that the surface is given as a mesh, and the curve is given as a polyline. The evaluation of the footpoint on a mesh surface can be done efficiently using a kd-tree [Bentley 1975] for the mesh faces. Such a kd-tree can be constructed in $O(n \log n)$ time, and enables query of the footpoint in $O(\log n)$ time, where n is the number of faces on S [de Berg et al. 2008]. Similarly, the evaluation of the footpoint on a polyline is done using a kd-tree for the segments of the polyline.

Sampling Points on Reference Shapes

Functional (3.5) is defined using a set of sample points of the reference shape. Our implementation allows the user to specify the number n of sample points. For a reference curve, the n sample points are generated by uniformly sampling the curve in the considered interval. For a reference surface, the sample points are generated according to [Alliez et al. 2003], which uniformly sample a mesh surface according to the given number of sample points.

Choosing Functional Weights

The functional weights in (3.2) need to be specified by the user. Note that each functional f is represented as sum of squared components σ_i

$$f = \sum_i (\sigma_i)^2.$$

A weight λ for functional f can be interpreted as a fixed contribution factor w_f for all components

$$\lambda f = \sum_i (w_f \sigma_i)^2,$$

which means

$$\lambda = N_f (w_f)^2$$

where N_f is the number of components in the definition of f. In this way, the weighted sum of a set of functionals $\left\{ f_i = \sum_j \left[\sigma_j^{(i)} \right]^2 \right\}$ can be written as

$$\sum_i \lambda_i f_i = \sum_i \sum_j \left[w_{f_i} \sigma_j^{(i)} \right]^2 .$$

The contribution factor w_{f_i} is a more intuitive way to measure the influence of a component in the target function. Our implementation allows the user to specify w_{f_i} for each functional f_i, instead of setting their weights $\{\lambda_i\}$ directly.

Note that the components of the geodesic property functional represent angular values, while the components of other functionals represent distance values. To combine them into a single target functional in a meaningful way, we normalize all components into compatible quantities as follows. Let D be the bounding box diameter for the initial web surface. If σ is a component representing a distance value, we divide it by D to obtain a ratio

$$\rho = \frac{\sigma}{D}. \tag{3.20}$$

For an angular component $\sigma = \sum_{k=0}^{2} \omega_{i+k} - \sum_{k=3}^{5} \omega_{i+k}$ in (3.17), we note that the following value is the discrete total geodesic curvature at vertex v for the polyline L containing e_i and e_{i+3} [Polthier and Schmies 1998]

$$\mathcal{K} = \frac{2\pi}{\theta} \cdot \frac{\sum_{k=0}^{2} \omega_{i+k} - \sum_{k=3}^{5} \omega_{i+k}}{2} = \frac{\sigma\pi}{\theta},$$

where $\theta = \sum_{k=1}^{5} \omega_{i+k}$. Also note that the part of L which is associated with v has length $(E_i + E_{i+3})/2$, with E_i and E_{i+3} being the length of e_i and e_{i+3}, respectively. Therefore, the geodesic curvature of L at v can be computed as

$$\kappa_g^L(v) = \frac{\mathcal{K}}{(E_i + E_{i+3})/2} = \frac{2\sigma\pi}{\theta(E_i + E_{i+3})}. \tag{3.21}$$

The ratio between bounding box diameter D and the radius of a circle with curvature $|\kappa_g^L(v)|$ is

$$\tau = D|\kappa_g^L(v)| = \frac{2D|\sigma|\pi}{\theta(E_i + E_{i+3})}. \tag{3.22}$$

τ equals the geodesic curvature of the polyline when the model is rescaled so that $D = 1$. Therefore, τ serves as a normalized geodesic curvature value. Since usually $\theta \approx 2\pi$, and $E_i + E_{i+3} \approx 2\overline{e}$ where $\overline{e}$ is the average edge length of the web surface, we approximate the signed version of ratio τ as

$$\rho = \frac{D\sigma}{2\overline{e}^{(0)}}, \tag{3.23}$$

where $\overline{e}^{(0)}$ is the average edge length of the initial web surface mesh. Now for each functional f_i, we can normalize its components $\{\sigma_j^{(i)}\}$ into ratio values $\{\rho_j^{(i)}\}$, according to (3.20) or (3.23). The contribution factor w_{f_i} for each functional is applied to these components to obtain the normalized target functional

$$F = \sum_i \sum_j \left[w_{f_i} \rho_j^{(i)} \right]^2. \tag{3.24}$$

3.2.9 Results

This section gives some examples of computing discrete webs by optimization.

Geodesic Webs

The following examples construct discrete geodesic webs for rationalization or form-finding. The initial mesh of the web is contained in the best approximation plane P for a set of sample points $\{q_i\}$ on the given constraint shapes, computed with PCA. This initial mesh is of rectangular shape, with its sides aligned with the two principal directions of $\{q_i\}$ which are parallel with P. The target functional is of the form

$$T = \lambda_{\text{prox}} f_{\text{prox,I}} + \lambda_{\text{fair}} \sum_i f_{\text{fair,II}}(\mathscr{L}_i) + \lambda_{\text{geod}} \sum_i f_{\text{geod}}(\mathscr{L}_i). \tag{3.25}$$

Figure 3.11 is a rationalization example. Figure 3.11(a) is the reference surface which needs to be approximated by the web. The initial mesh is shown in Figure 3.11(b). Figure 3.11(c) is the optimized mesh, which covers the whole reference surface. Figure 3.11(e) is the color-coded normalized geodesic curvature τ (see Equation (3.22) for definition) for the three families of polylines (shown in white) on the optimized mesh. Since we are

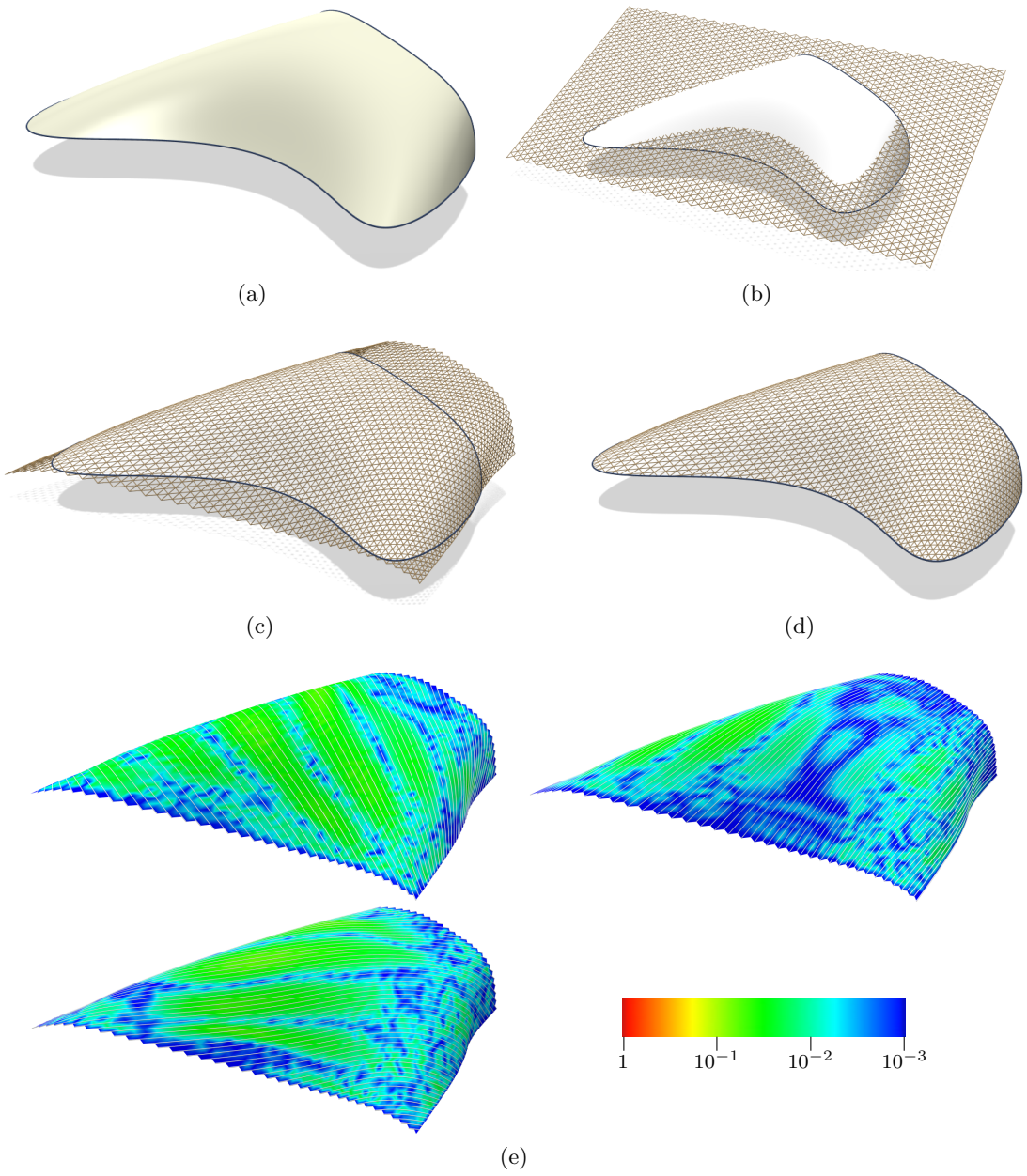


Figure 3.11: Rationalization using a geodesic web. (a) The reference surface. (b) The initial mesh for the web. (c) The optimized mesh. (d) The polylines of the optimized mesh are trimmed by the reference surface, resulting in a curve network on the reference surface. (e) The color-coded normalized geodesic curvature values (defined in Equation (3.22)) for the three families of mesh polylines (in white).

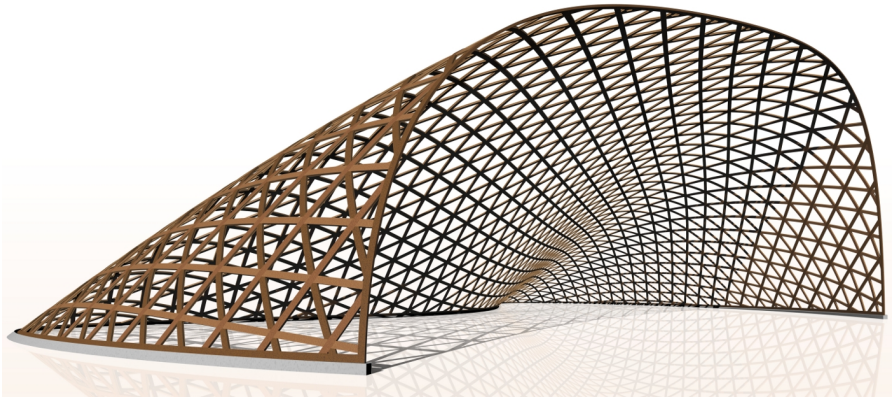


Figure 3.12: An architectural design based on Figure 3.11(d), where the curve elements can be realized by bending straight wooden panels.

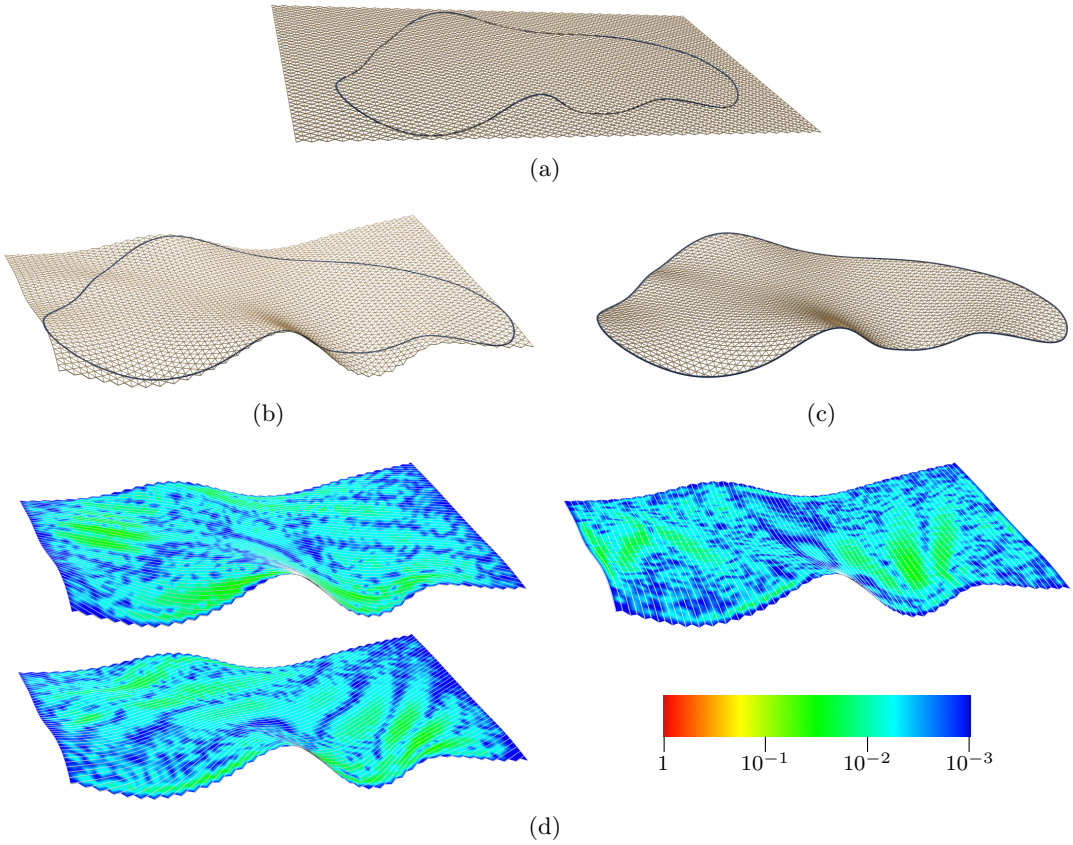


Figure 3.13: Form-finding using a geodesic web. (a) The given constraint curve (in dark blue), and the initial mesh for the web. (b) The optimized mesh, which is incident with the constraint curve. (c) The polylines of the optimized mesh are trimmed by the constraint curve. (d) The color-coded normalized geodesic curvature values for the three families of mesh polylines.

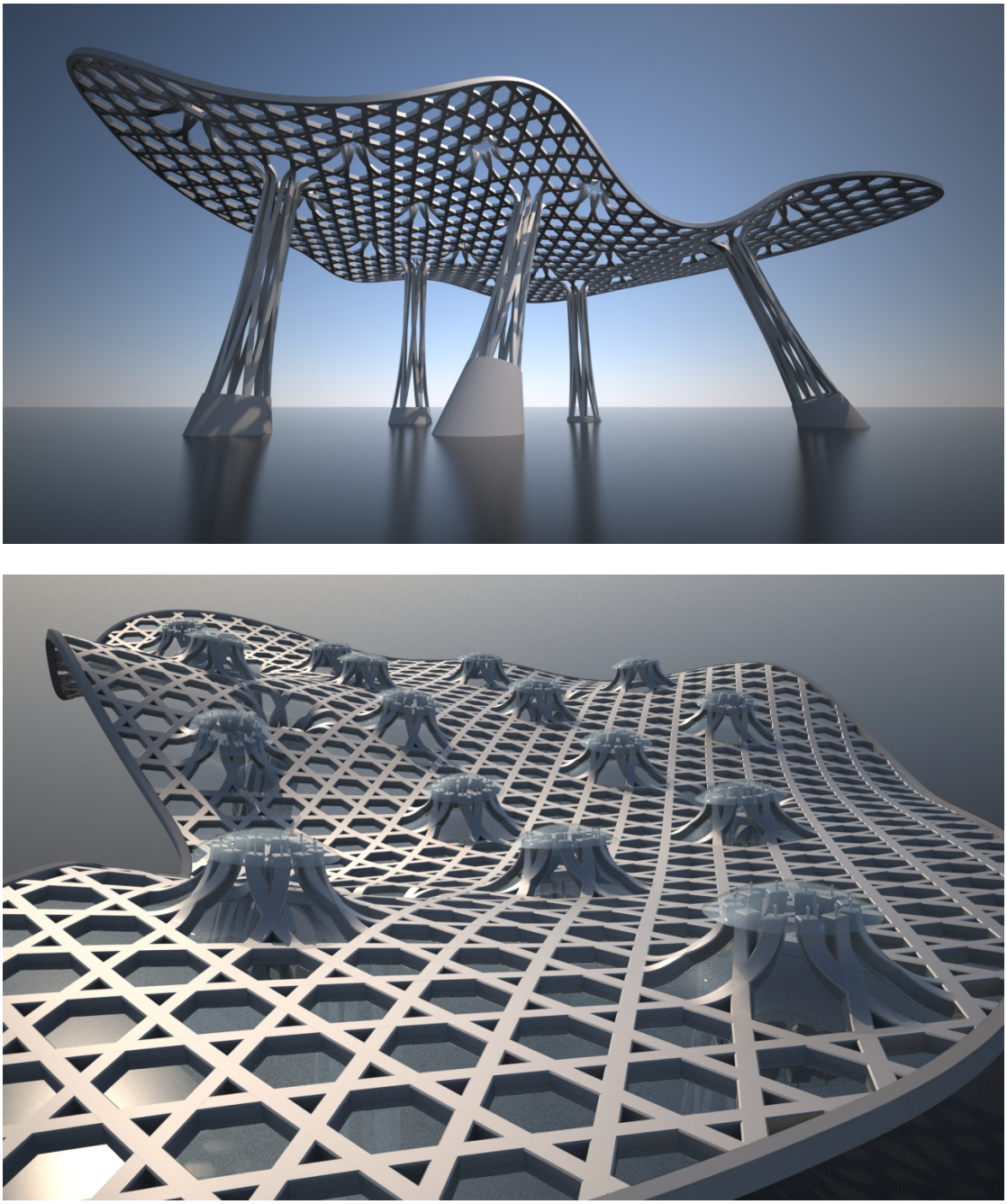


Figure 3.14: By deleting some polylines in the geodesic web of Figure 3.13(c) we obtain a tri-hex structure which serves as the basis of an architectural design. The greater part of beams in this example is to be made from layers, each individually bent into shape (which is possible due to the geodesic property).

only concerned with the curves on the given surface, we trim the optimized mesh with the boundary of the given surface, and obtain a curve network shown in Figure 3.11(d). Figure 3.12 is an architectural design based on the geodesic polylines in Figure 3.11(d), which can be realized by bending wooden panels.

Figure 3.13 shows an example of form-finding. We are given a curve which represents the desired boundary of the geodesic web. The geodesic web needs to be incident with this curve. Figure 3.13(a) shows the given curve (in blue) and the initial mesh. Figure 3.13(b) is the optimized mesh. Note that the second order difference fairing functional $f_{\text{fair,II}}$ in (3.25) leads to shrinking of the mesh surface, while the proximity functional $f_{\text{prox,I}}$ prevents the mesh from shrinking beyond the given constraint curve. Figure 3.13(d) shows the normalized geodesic curvatures for the three polyline families in the optimized mesh. We trim the optimized mesh with the given boundary curve to obtain a curve network shown in Figure 3.13(d). An architectural design based on the curves in Figure 3.13(d) is shown in Figure 3.14. Here every second curve of each family is removed to produce a tri-hex structure.

Planar Webs

We show below two examples of rationalization using planar webs. The target functional is of the form

$$F = \lambda_{\text{prox}} f_{\text{prox}} + \lambda_{\text{fair}} \sum_i f_{\text{fair,III}}(\mathscr{L}_i) + \lambda_{\text{planar}} \sum_i f_{\text{planar}}(\mathscr{L}_i). \tag{3.26}$$

In Figure 3.15, the reference surface is the top of the Lilium Tower. We use $f_{\text{prox,II}}$ as the proximity functional in (3.26). The vertices of the initial mesh is incident with the reference surface (see Figure 3.15(a)). Since we are interested in non-trivial planar webs, we fix the auxiliary variables of the planar property for four polylines, such that they represent four planes which do not have a common point. This prevents the mesh from evolving into a shape where all polylines are in planes with a common intersection point. The optimized mesh is shown in Figure 3.15(b). To visualize the deviation of a polyline L from planar curves, we compute a best fitting planar polyline $\overline{L}$ of L for comparison. The vertices of $\overline{L}$ are the footpoints of the vertices of L on a plane $\overline{P}_L$, which is the best fitting plane for the vertices of L. In Figure 3.15(b), the polylines of the optimized mesh are shown in brown, while their best fitting planar polylines are shown in dark blue with smaller radius (75% of the radius of the brown polylines). In this way, a large deviation of the optimized polylines from the planar property is indicated by the visibility of blue polylines. Figure 3.15(b) shows that the planarity constraints for the polylines are well satisfied. To check whether or not the planar web in Figure 3.15(b) is trivial, we can search for a point p which is closest to all planes $\{P_i\}$ for the polylines, by minimizing the following target function

$$T(p) = \sum_i [\text{dist}(p, P_i)]^2, \tag{3.27}$$

where $\text{dist}(p, P_i)$ is the distance from point p to plane P_i. This is a linear least squares problem. After that we compute the footpoint of p on each plane P_i. If the planes

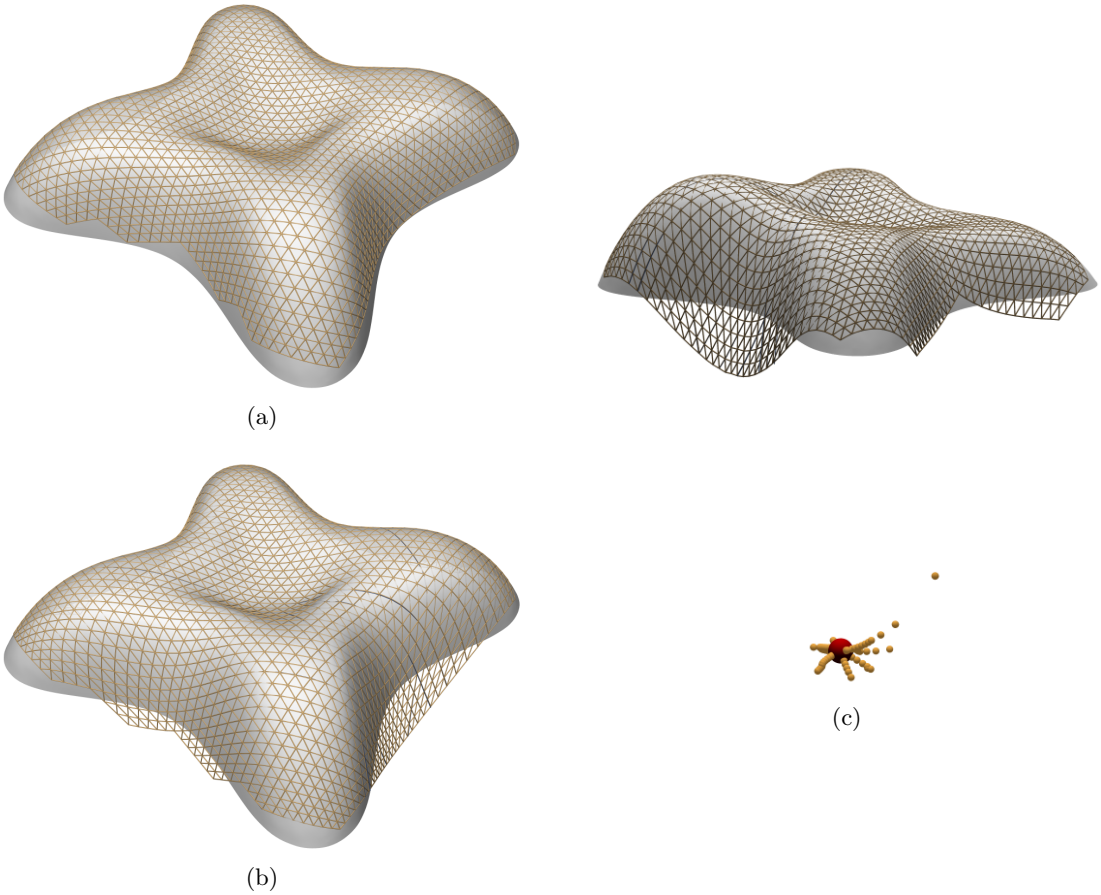


Figure 3.15: Rationalization using a planar web. (a) The reference surface and the initial mesh. (b) The optimized mesh, with the best fitting planar curves (in dark blue, with smaller radius) for the mesh polylines. The fact that the dark curves are hardly visible indicates small deviation between the polylines and exactly planar curves. (c) We check whether the planar web is trivial, using a point p (in red) minimizing the target function (3.27) and its footpoint (in yellow) onto the planes containing the polylines. The footpoints do not concentrate at point p, indicating that the web is not trivial.

$\{P_i\}$ have a common intersection point, the footpoints will be concentrated at p. Figure 3.15(c) shows the point p (in red) and the footpoints (in yellow) for the optimized mesh. We can see that the computed planar web is not trivial.

In Figure 3.16, we aim at a web of planar curves with tri-hex connectivity. The initial mesh in Figure 3.16(b) is created using 3D modeling tools. We require every second polyline of each family to be planar. For the proximity functional we use $f_{\text{prox,I}}$, which leads to an optimized mesh that covers the whole reference surface (see Figure 3.16(c)). By removing the mesh surface regions outside the reference surface, and by collecting the polylines with planarity constraints, we obtain a tri-hex structure shown in Figure

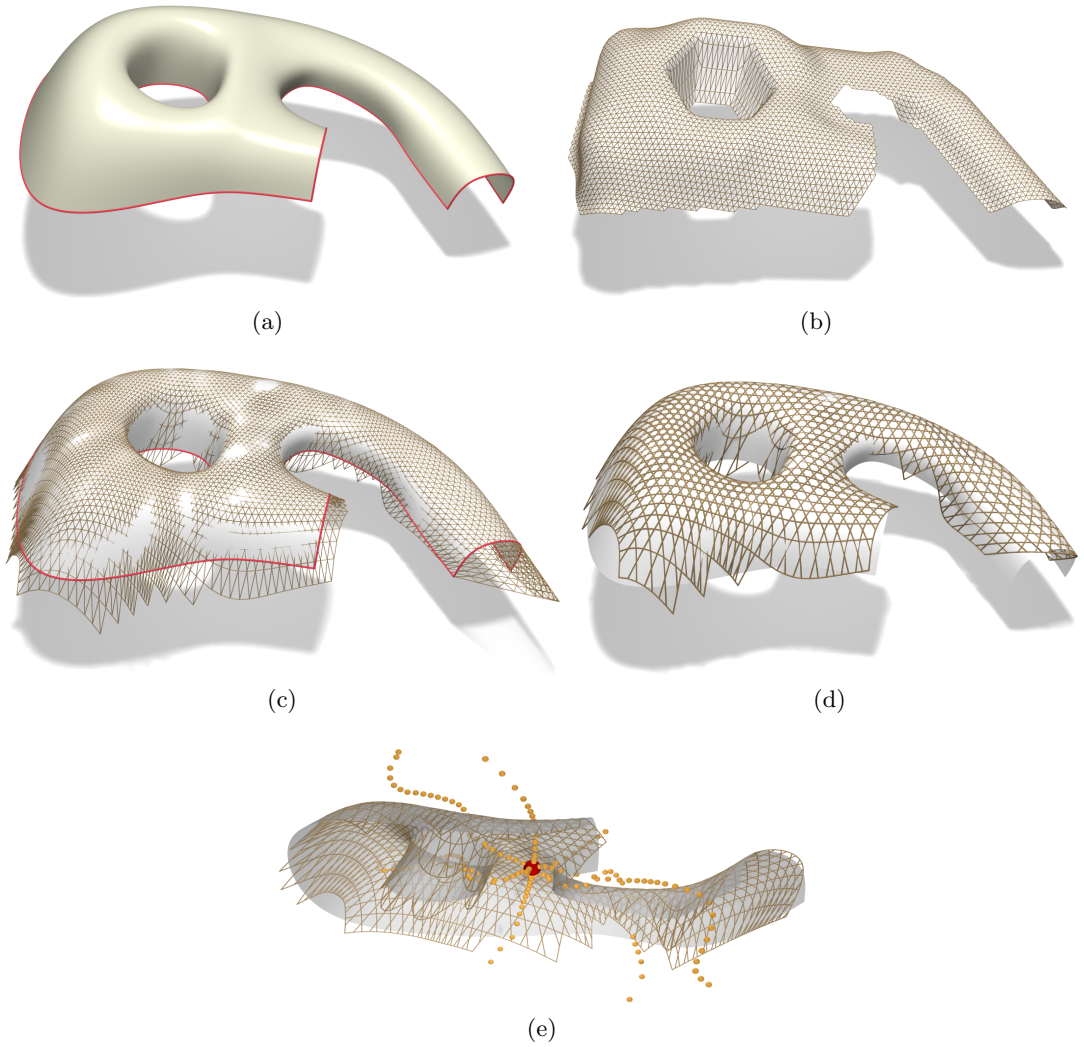


Figure 3.16: Rationalization using planar curves with tri-hex connectivity. (a) The reference surface. (b) The initial mesh, with the planarity constriants imposed on every second polyline of each family. (c) The optimized mesh. (d) By deleting the mesh region outside the range of the reference surface, and collecting the polylines with planarity constraints, we obtain a curve network of tri-hex connectivity. The fitting planar curves for the polylines are shown in dark blue with smaller radius, indicating small deviation between the polylines and planar curves. (e) We check whether this planar web is trivial, in a way similar to Figure 3.15(c). The deviation between the red point and the yellow points indicate that the web is not trivial.

3.16(d). In Figure 3.16(d), the best fitting planar polylines are shown with smaller radius in dark blue, indicating only small deviation between them. Figure 3.16(e) shows the point p (in red) which minimizes (3.27), and the footpoints of p onto the planes. We see that the computed planar web is not trivial. Note, however, that this planar web may not be useful, due to aesthetic reasons. This example shows that it can be difficult to find a web which satisfies the imposed conditions and is aesthetically pleasing at the same time.

Webs with Circular Curves

Figure 3.18 shows a discrete web with some curves being circular. We want this web to approximate a given surface, with two families of geodesics and one family of circular curves which are in vertical planes. From the analysis in Section 3.2.7, we see that if we impose geodesic and circular conditions on every polyline within the corresponding families, we may not have enough degrees of freedom. Therefore, we only require every 4th polyline of the circular family to be a circular polyline in a vertical plane. Besides, to reduce the number of constraints for the web surface, we do not use the whole given surface as reference shape, but take the boundary curve and another curve on the surface as constraint curves (see Figure 3.18(a)). Let $\mathscr{L}_1$, $\mathscr{L}_2$ be the geodesic families of polylines, and $\mathscr{L}_3$ be the circular family. We use the following target functional

$$\begin{aligned} F &= \lambda_{\text{prox}} f_{\text{prox,I}} + \lambda_{\text{geod}} \sum_{i=1,2} f_{\text{geod}}(\mathscr{L}_i) + +\lambda_{\text{circle}} f_{\text{circle}}(\mathscr{L}_3) \\ &\quad +\lambda_{\text{planar}} f_{\text{planar}}(\mathscr{L}_3) + \lambda_{\text{fair,II}} \sum_{i=1,2} f_{\text{fair,II}}(\mathscr{L}_i) + \lambda_{\text{fair,III}} f_{\text{fair,III}}(\mathscr{L}_3). \end{aligned}$$

Here the functional f_{planar} is used to constrain the circular polylines to lie in vertical planes. The vertices of the initial mesh lie on the given surface (see Figure 3.18(b)). Figure 3.18(c) shows the optimized mesh, as well as the best fitting vertical circular curves to the polylines with circularity constraints. The fitting circular curves are shown in dark blue with larger radius (130% of the radius of the brown polylines). Figure 3.18(d) shows the optimized mesh with the constraint curves. Figure 3.18(e) shows the normalized geodesic curvature plots for the two geodesic families of polylines. We trim the optimized mesh by a plane which represents the ground, to obtain a curve network useful for architecture (see Figure 3.18(f)). Figure 3.18 is an architectural design based on this curve network.

Webs of Cylinder Topology

Next we show some webs of cylinder topology. Such a web can represent the shape of a tower. Among the three families of polylines, one of them contain close polylines, which we will call the *horizontal* family. The remaining two families will be called the *diagonal* families. The regular connectivity of the mesh means that all polylines in the horizontal family have the same number of vertices. In the examples below, all polylines in the diagonal families are required to be geodesics, and some polylines of the horizontal

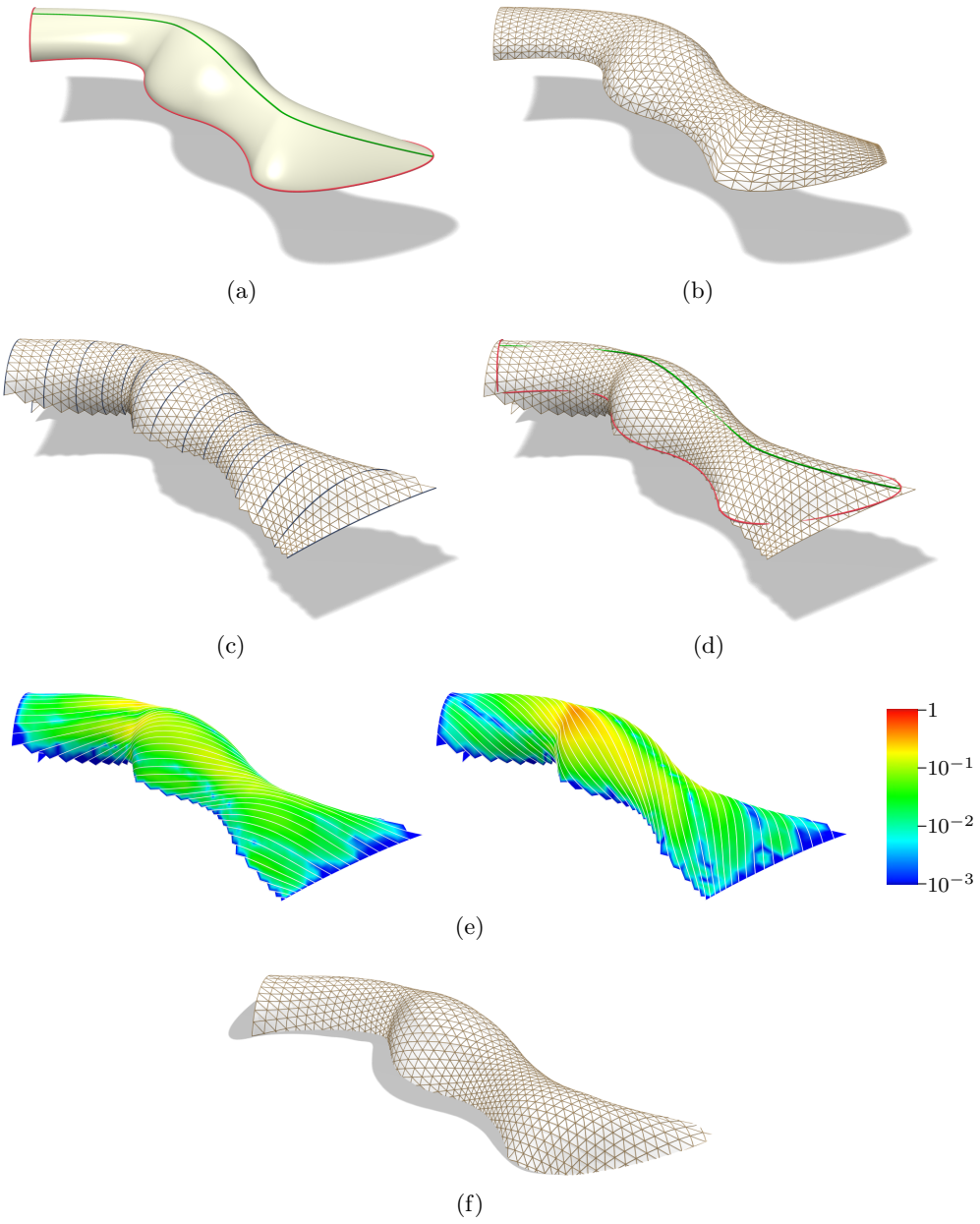


Figure 3.17: A web containing circular curves. (a) The boundary curve (in red) and another curve (in green) on a given surface are taken as the constraint curves. (b) The initial mesh for the web. (c) The optimized mesh. The best fitting vertical circular curves are shown in dark blue with larger radius, indicating small deviation from the respective mesh polylines. (d) The optimized mesh is indeed close to the constraint curves. (e) The normalized geodesic curvature for the two families of geodesic polylines. (f) The polylines are trimmed by a plane representing the ground, resulting in a curve network useful for architectural design (see Figure 3.18).

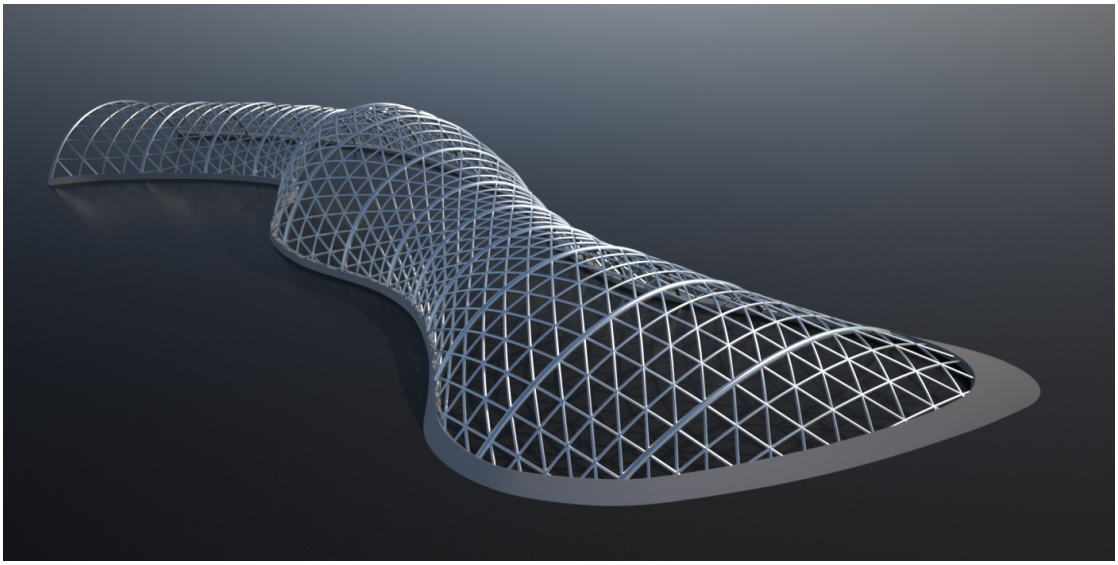


Figure 3.18: An architectural design based on Figure 3.17(f).

families are required to be planar curves in horizontal planes. Such conditions are useful in constructing the outer surface of a building by two families of long strips, while requiring the intersections of these strips to be aligned with floor levels of the building (see Figure 3.4 for an example). Let $\mathscr{L}_1$, $\mathscr{L}_2$ be the diagonal families of polylines, and $\mathscr{L}_3$ be the horizontal family. The target functional is of the form

$$\begin{aligned} F &= \lambda_{\text{prox}} f_{\text{prox}} + \lambda_{\text{bdry}} f_{\text{bdry}} + \lambda_{\text{geod}} \sum_{i=1,2} f_{\text{geod}}(\mathscr{L}_i) \\ &\quad + \lambda_{\text{planar}} f_{\text{planar}}(\mathscr{L}_3) + \lambda_{\text{fair,II}} \sum_{i=1,2} f_{\text{fair,II}}(\mathscr{L}_i) + \lambda_{\text{fair,III}} f_{\text{fair,III}}(\mathscr{L}_3). \end{aligned} \tag{3.28}$$

Figure 3.19 is a rationalization example. The two boundary curves of the given reference surface lie in horizontal planes (see Figure 3.19(a)). We use $f_{\text{prox,II}}$ as the shape proximity functional f_{prox}. And the boundary proximity functional f_{bdry} penalizes the deviation between the boundary curves of the web and the boundary curves of the given surface. In order to have enough degrees of freedom for the proximity constraints, we only require every 6th polyline in the horizontal family to be in a horizontal plane. In the initial mesh, all vertices lie on the given surface, and all polylines of the horizontal family lie on equidistant horizontal planes (see Figure 3.19(b)). Figure 3.19(c) shows the optimized mesh, as well as the best horizontal fitting curves for the polylines with planarity constraints. The horizontal fitting curves are shown in dark blue with larger radius (130% of the radius of the brown polylines), indicating small deviation of the polylines from horizontal curves. The normalized geodesic curvature for the diagonal families are shown in Figure 3.19(d). Figure 3.20 is an architectural design based on this web.

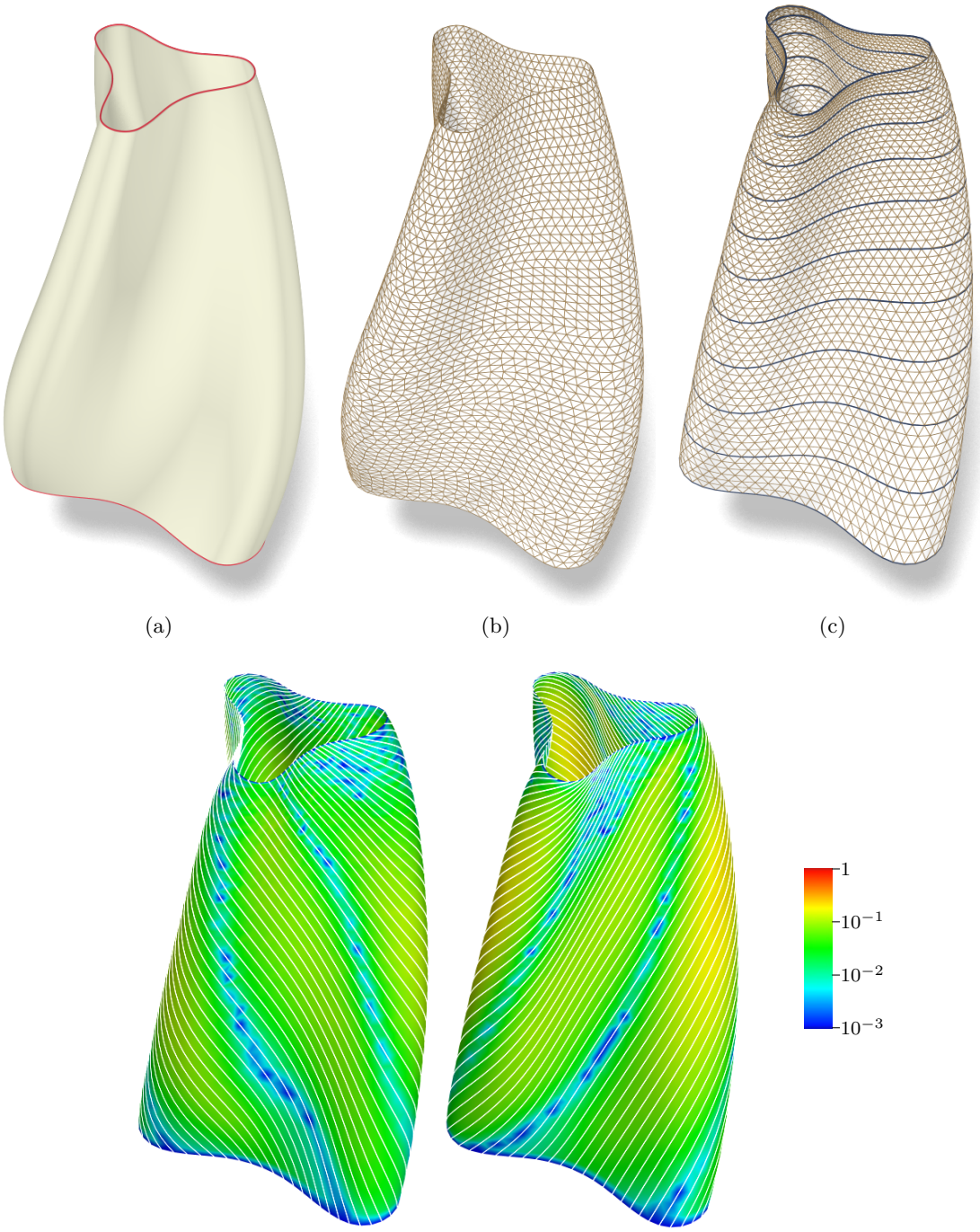


Figure 3.19: Rationalization using a web of cylinder topology. (a) The reference surface. (b) The initial mesh for the web. (c) The optimized mesh. The fitting horizontal planar curves are shown in dark blue with larger radius, and indicate small deviation from the corresponding polylines. (d) The normalized geodesic curvature for the two families of geodesic polylines.

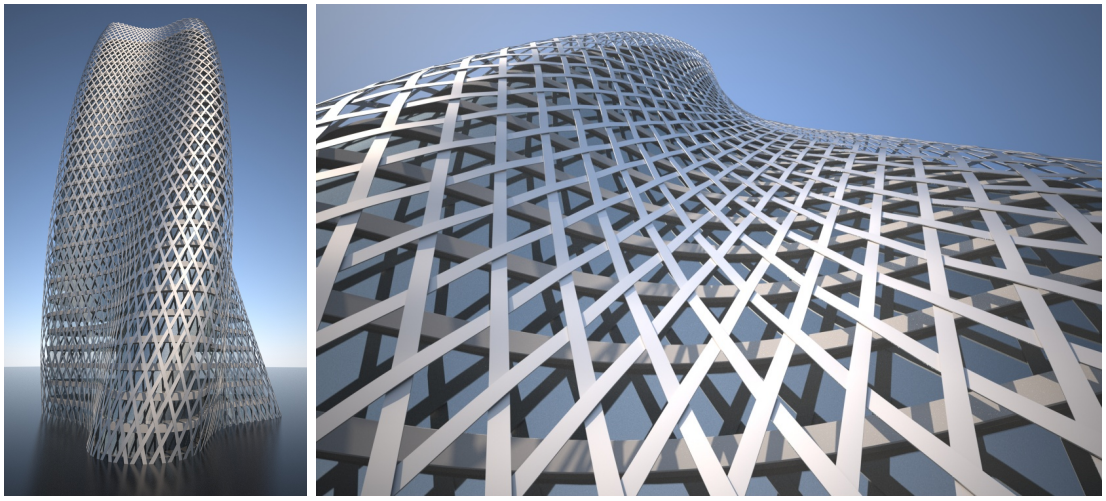


Figure 3.20: An architectural design based on the web in Figure 3.19. For the diagonal families of curves, we introduce additional high frequency bending of small amplitude to obtain a "weaving" design.

Figure 3.22 is a form-finding example. The two boundary curves of the web are constrained to two ellipses in horizontal planes (shown in Figure 3.22(a)). We require all polylines in the horizontal families to be horizontal planar curves. It can be shown that there are enough degrees of freedom for such conditions: Every boundary vertex of the mesh is constrained on a curve, and has one d.o.f. for its position. Each interior vertex has three d.o.f. for its position. Therefore, the total number of d.o.f. for vertex positions of the mesh is $3N_{IV} + N_{BV}$, where N_{IV} and N_{BV} are the number of interior vertices and boundary vertices, respectively. The geodesic conditions induce $2N_{IV}$ constraints. The horizontal planar condition for each interior polyline L (i.e., with no boundary vertex) of the horizontal family induces $N_V^L - 1$ constraints, with N_V^L being the number of vertices on L. In total the planar conditions induce $\sum(N_V^L - 1) = N_{IV} - N_h$ constraints for the interior polylines of the horizontal family, where N_h is the number of interior polylines. So the number of the remaining d.o.f. is

$$3N_{IV} + N_{BV} - 2N_{IV} - (N_{IV} - N_h) = N_{BV} + N_h > 0$$

This form-finding problem is solved in a multiresolution way: We start from a coarse initial mesh M_0, with its boundary vertices incident with the constraint curves (see Figure 3.22(b)). The target functional (3.28) is minimized to obtain an optimized mesh $\widetilde{M}_0$ (see Figure 3.22(c)). Then $\widetilde{M}_0$ is subdivided by introducing new vertices at midpoints of mesh edges (see Figure 3.21), which gives a finer new mesh M_1 as the initial mesh for the next level of optimization (see Figure 3.22(d)). Such subdivision and optimization is iterated, until we obtain an optimized mesh at the desired resolution. Figure 3.22(e) shows the optimized mesh $\widetilde{M}_2$. Also shown in Figure 3.22(e) are the horizontal planar fitting curves in dark blue with smaller radius, which are not visible. Figure 3.22(f)

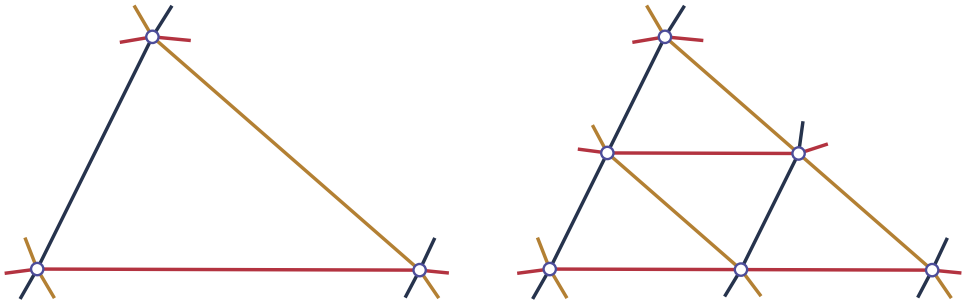


Figure 3.21: Subdividng a regular mesh by connecting the mid-points of edges.

shows the normalized geodesic curvature for the polylines in the diagonal families. We can see that both the planarity conditions and the geodesic conditions are well satisfied, confirming our analysis for the degrees of freedom.

We can even iteratively "grow" a web of cylinder topology, using a set of *guiding curves* $\{\mathcal{C}_i\}$ to control the shape of the surface. In each iteration, we optimize a web of cylinder topology with respect to the following constraints: The bottom boundary vertices are constrained to be close to a given curve in the plane $z = 0$. The top boundary curve is constrained to be in a horizontal plane $z = (k-1)h$, where h is a constant and k is the number of polylines in the horizontal family. We require all interior polylines of the horizontal family to be in horizontal planes. Besides, we sample the guiding curves between the planes $z = 0$ and $z = (k-1)h$ to obtain a set of points $\{q_j\}$, and the mesh need to be close to these points. Note that the constraints for polylines are the same as the previous example, except that the top boundary vertices are not constrained to a curve, but to a horizontal plane. This leads to one more degree of freedom for each top boundary vertex. From an analysis similar to the previous example, we conclude that the difference between the number of d.o.f. for the vertex coordinates and the number of constraints for the polylines equals $\frac{3}{2}N_{BV} + N_h > 0$. Therefore, it is possible to have enough d.o.f. for the positional constraints induced by the points $\{q_j\}$. The optimized mesh in each iteration is augmented by one layer of new vertices to become the initial mesh for the next iteration: Let $\widetilde{M}^{(k)}$ be the optimized mesh for the current iteration, with k polylines in its horizontal family. We project the top boundary vertices of $\widetilde{M}^{(k)}$ onto the plane $z = kh$ to obtain a set of new vertices. These new vertices are added to $\widetilde{M}^{(k)}$ to generate a new mesh $M^{(k+1)}$ of cylinder topology, with $k+1$ polylines in the horizontal family. $M^{(k+1)}$ becomes the initial mesh for the next iteration. Figure 3.23 is an example of such a growing web. Figure 3.23(a) shows the bottom boundary constraint curve (in green) and the guiding curves (in red). We start from an initial mesh $M^{(3)}$ with 3 polylines in the horizontal family, which is shown in Figure 3.23(b). Figure 3.23(c) is the optimized mesh $\widetilde{M}^{(3)}$. Figure 3.23(d) shows the initial mesh $M^{(4)}$ for the next iteration, which is derived from $\widetilde{M}^{(3)}$. Figure 3.23(e) is the optimized mesh $\widetilde{M}^{(38)}$, with the horizontal planar fitting curves shown in dark blue with smaller radius (not visibility in the figure). Figure 3.23(f) shows the normalized geodesic curvature values for the diagonal families in $\widetilde{M}^{(38)}$.

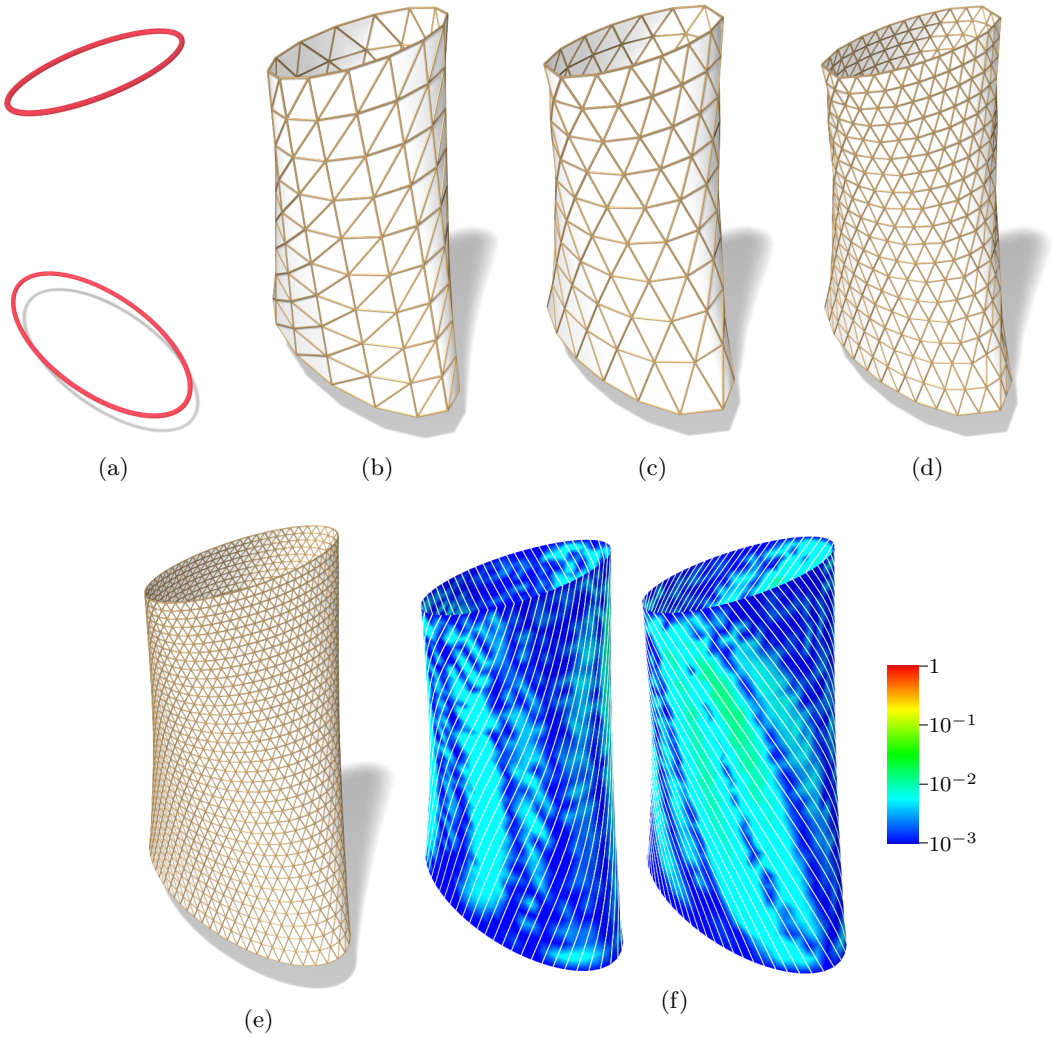


Figure 3.22: Form-finding using a web of cylinder topology, in a multiresolution way. (a) The constraint curves. (b) The initial mesh M_0 for the initial level of resolution. (c) The optimized mesh $\widetilde{M_0}$ for M_0. (d) $\widetilde{M_0}$ is subdivided to produce the initial mesh M_1 for the next level of resolution. (e) The optimized mesh $\widetilde{M_2}$. The fitting horizontal planar curves are shown in dark blue (which are not visible, indicating the planarity of the corresponding polylines). (f) The normalized geodesic curvature for the diagonal families of polylines in $\widetilde{M_2}$.

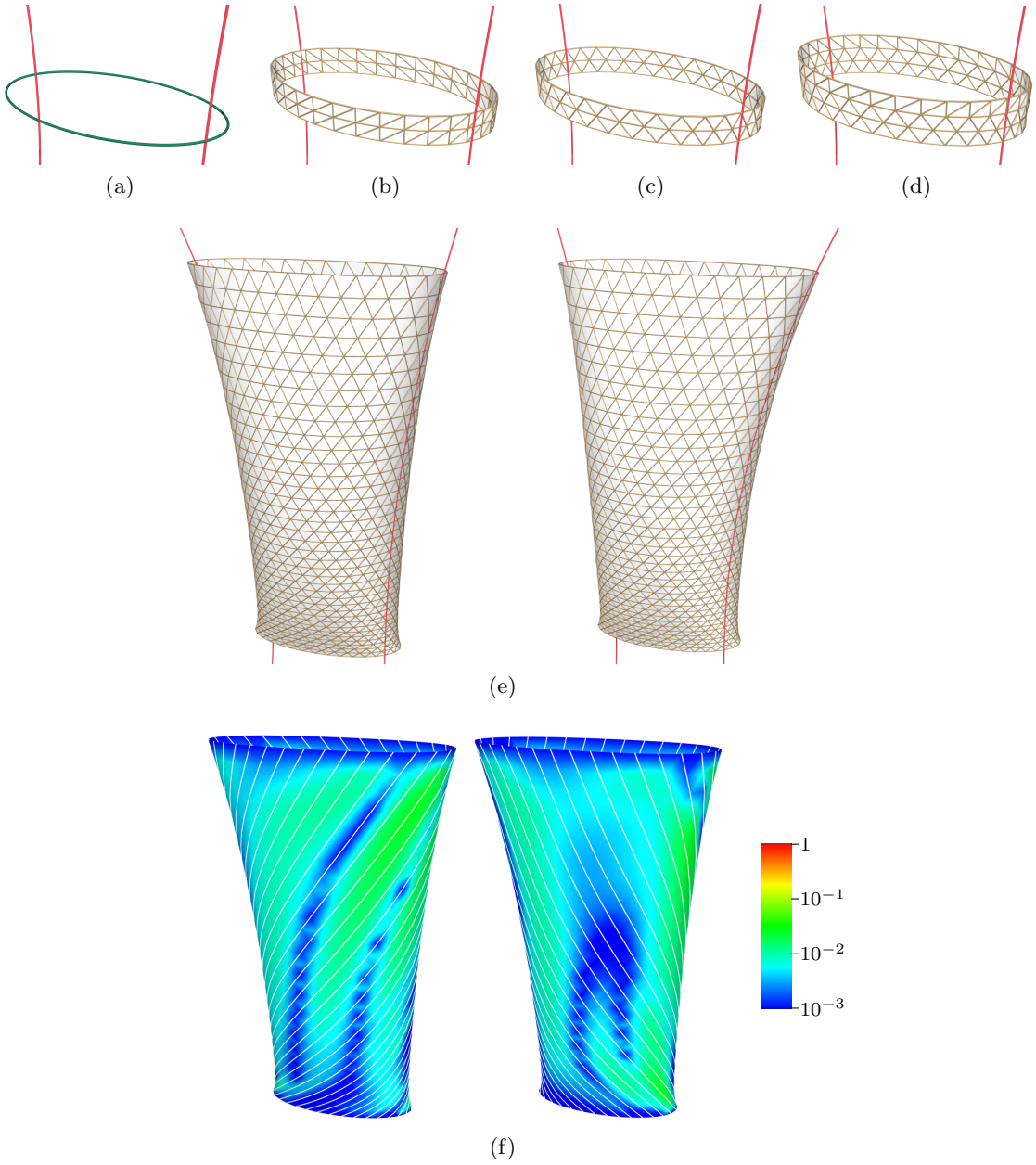


Figure 3.23: Growing a web of cylinder topology. (a) The bottom constraint curve (in green) and the guiding curves (in red). (b) The initial mesh $M^{(3)}$. (c) The optimized mesh $\widetilde{M}^{(3)}$ for $M^{(3)}$. (d) $\widetilde{M}^{(3)}$ is augmented by a layer of new vertices to generate the initial mesh $M^{(4)}$ for the next iteration of optimization. (e) The optimized mesh $\widetilde{M}^{(38)}$. The fitting horizontal planar curves are shown in dark blue (which are not visible, indicating that the horizontal planar conditions are well satisfied). (f) The normalized geodesic curvature for the diagonal families of polylines in $\widetilde{M}^{(38)}$.

Weights and Timing

Table 3.1 shows the weights and timing data for some examples above. Here $\#V$ is the number of variables for the optimization problem. T is the timing (in seconds) for the optimization, measured on a laptop with a 2.4GHz CPU. The m's are the maximum magnitude of the normalized components ρ in target functional (3.24) for specific properties of the optimized mesh, while the w's are the corresponding contribution factors used in (3.24).

Fig	$\#V$	T	m_{geod}	w	m_{planar}	w	m_{circle}	w	$m_{\text{fair,II}}$	w	$m_{\text{fair,III}}$	w	m_{bdry}	w	m_{prox}	w
3.11	7563	100	0.06	0.0025					0.002	0.1			$1{\cdot}10^{-4}$	1	$2{\cdot}10^{-4}$	1
3.13	10074	214	0.05	0.09					0.001	1			0.001	1		
3.15	5155	66			0.009	3					0.01	0.1			0.002	1
3.16	13396	1002			0.01	3					0.01	1	0.009	1	0.009	1
3.17	12537	210	0.4	0.045	0.003	1.5	0.002	1.5	0.002	4	0.002	1	0.006	0.4	0.005	0.4
3.19	13187	176	0.2	0.09	0.01	1.5			0.005	3	0.003	1	0.004	1.5	0.04	1

Table 3.1: Weights and timing for the optimization of discrete webs.

3.3 Exact Planar Webs

For the discrete webs constructed by optimization, usually the constraints for mesh polylines can only be satisfied up to a certain threshold. For planar webs, however, it turns out to be easy to give a complete parametric description of them. Such a description enables us to compute a planar web using three families of planes, and the curves in this web will be exactly planar. In this section we discuss the construction and modification of these exact planar webs.

3.3.1 Discrete Planar Webs from Plane Families

Assume that the three families $\mathscr{L}_1$, $\mathscr{L}_2$, $\mathscr{L}_3$ of mesh polylines in a regular triangle mesh

$$\begin{aligned} \mathscr{L}_1 &= \{L_1(i)\}_{i\in\mathbb{Z}}, \\ \mathscr{L}_2 &= \{L_2(j)\}_{j\in\mathbb{Z}}, \\ \mathscr{L}_3 &= \{L_3(k)\}_{k\in\mathbb{Z}} \end{aligned}$$

are defined such that polylines $L_1(i)$, $L_2(j)$ and $L_3(k)$ intersect in a vertex v_{ij} if and only if $i + j + k = c$ where $c \in \mathbb{Z}$ is a constant. In other words, the polylines can be represented as

$$\begin{aligned} L_1(i) &= \{\ldots, v_{i0}, v_{i1}, v_{i2}, \ldots\}, \\ L_2(j) &= \{\ldots, v_{0j}, v_{1j}, v_{2j}, \ldots\}, \\ L_3(k) &= \{\ldots, v_{-k,c}, v_{1-k,c-1}, v_{2-k,c-2}, \ldots\}. \end{aligned}$$

For a planar web, each polyline $L_r(i)$ is contained in a plane $P_r(i)$ $(r = 1, 2, 3)$. Therefore, there are three families of planes $\{P_1(i)\}_{i\in\mathbb{Z}}$, $\{P_2(j)\}_{j\in\mathbb{Z}}$, $\{P_3(k)\}_{k\in\mathbb{Z}}$ such that the mesh vertices arise as the intersection

$$v_{ij} = P_1(i) \cap P_2(j) \cap P_3(k), \quad \text{where } i + j + k = c. \tag{3.29}$$

So the general description of a discrete planar web is as follows: *Select three families of planes and define the vertices of a triangle mesh by* (3.29). Note that by this construction, each mesh polyline is exactly planar. Figure 3.24 shows an example where all three families are linear pencils of planes.

3.3.2 Continuous Planar Webs

The above method can be extended to describe a smooth surface where a web of planar curves exists: Take three families of planes $\{P_1(u)\}_{u\in\mathbb{R}}$, $\{P_2(v)\}_{v\in\mathbb{R}}$, $\{P_3(w)\}_{w\in\mathbb{R}}$, and define a surface by

$$S(u, v) = P_1(u) \cap P_2(v) \cap P_3(w) \quad \text{where } u + v + w = c, \tag{3.30}$$

where $c \in \mathbb{R}$ is a constant. Then the iso-parameter lines $u = \text{const}$, $v = \text{const}$ and $w = \text{const}$ define planar curves in this surface. On such a surface, we can recover three families of parametric planar curves $\{C_1(i)\}_{i\in\mathbb{Z}}$, $\{C_2(j)\}_{j\in\mathbb{Z}}$, $\{C_2(k)\}_{k\in\mathbb{Z}}$ with the same connectivity as the planar polylines in a discrete planar web, where

$$\begin{aligned} C_1(i) &= S(\alpha i, r), \quad r \in \mathbb{R}, \\ C_2(j) &= S(s, \alpha j), \quad s \in \mathbb{R}, \\ C_3(k) &= S(t, c - t - \alpha k), \quad t \in \mathbb{R} \end{aligned} \tag{3.31}$$

with $\alpha \in \mathbb{R}$ being a non-zero constant. We call the surface in (3.30) a *continuous planar web*. By choosing different values of α, we can obtain planar curve networks of different density on the same continuous planar web (see Figure 3.25).

3.3.3 Constructing Continuous Planar Webs

To construct a continuous planar web, we need to select three continuous families of planes. A plane can be identified with a normal vector n and a scalar d, such that any point p on the plane satisfies equation $n \cdot p = d$. Therefore, a continuous one-parameter family of planes $P(t)$ can be represented by a vector-valued function $n(t)$ and a scalar-valued function $d(t)$. We use the notation $P(t) = (d(t), n(t))$ for this representation. For three families of planes $P_1(u) = (d_1(u), n_1(u))$, $P_2(v) = (d_2(v), n_2(v))$, $P_3(w) = (d_3(w), n_3(w))$, the surface in (3.30) has a parametric form

$$S(u, v) = \frac{d_1(u)(n_2(v) \times n_3(w)) + d_2(v)(n_3(w) \times n_1(u)) + d_3(w)(n_1(u) \times n_2(v))}{\det(n_1(u), n_2(v), n_3(w))}, \tag{3.32}$$

where $w = c - u - v$.

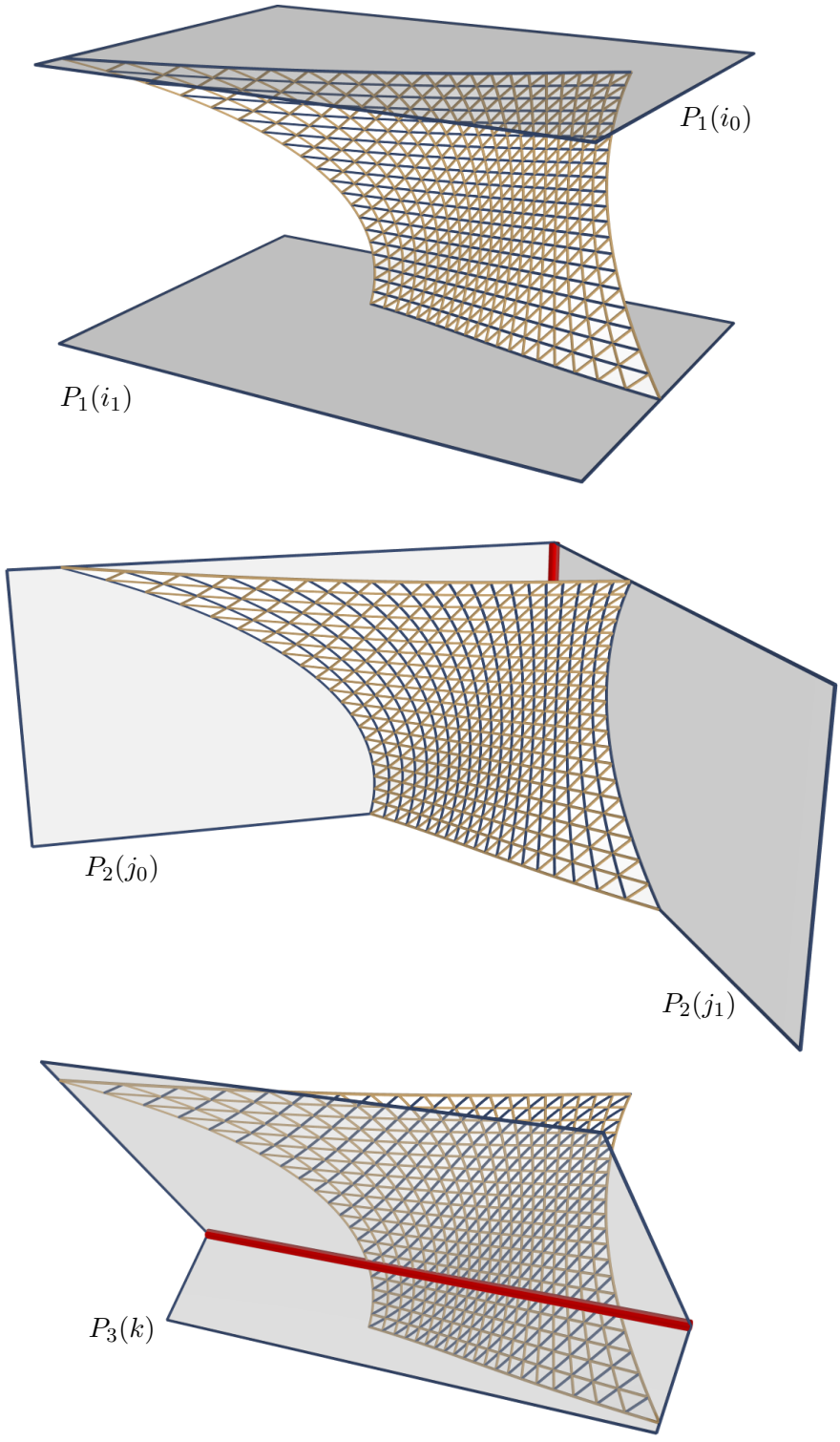


Figure 3.24: The vertices v_{ij} of a planar web are generated by the families P_1, P_2, P_3 of planes via $v_{ij} = P_1(i) \cap P_2(j) \cap P_3(c - i - j)$. Here all three families of planes are linear pencils.

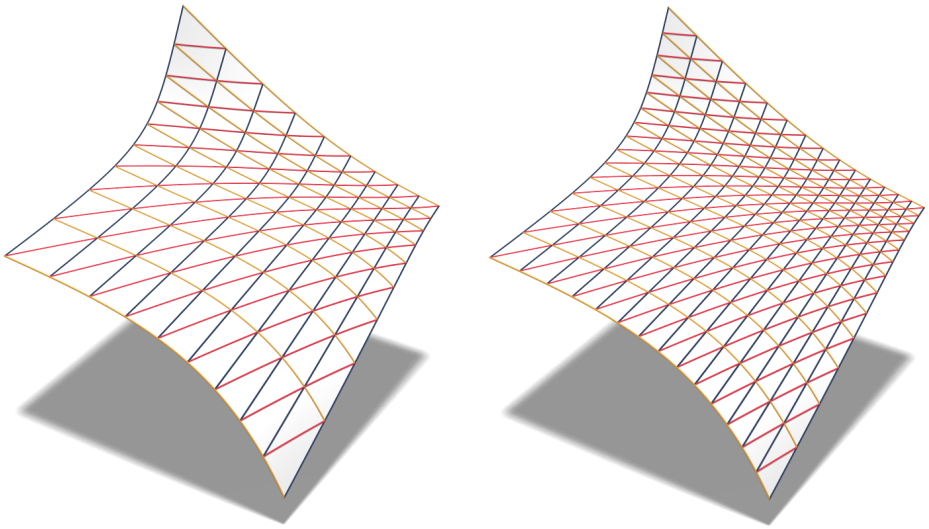


Figure 3.25: On a same surface defined by (3.30), we can obtain webs of planar curves of different density, by choosing different values of α in (3.31).

Here we present two approaches to select plane families $P_1(u)$, $P_2(v)$, $P_3(w)$. In the first approach, the three families of planes are orthogonal to the three coordinate axes respectively. In the second approach, each family consists of the normal planes of a parametric curve.

Plane Families Orthogonal to Coordinate Axes

If the three families of planes $P_1(u)$, $P_2(v)$, $P_3(w)$ are orthogonal to x axis, y axis and z axis, respectively, they have constant normal vectors

$$\begin{aligned} n_1(u) &= (1,0,0), \\ n_2(v) &= (0,1,0), \\ n_3(w) &= (0,0,1), \end{aligned}$$

and (3.32) is simplified to

$$S(u,v) = (d_1(u), d_2(v), d_3(c-u-v)). \tag{3.33}$$

In this case, we can control the shape of $S(u,v)$ simply using the three scalar functions $d_1(u), d_2(v), d_3(w)$. Here we give some examples of such surfaces. In all these examples, we use $c = 2$ in Equation (3.33), and we are interested in the region of $S(u,v)$ corresponding to the parameter domain $0 \le u, v \le 1$, meaning that $0 \le c-u-v \le 2$. Therefore, we only need to specify the functions $d_1(u)$, $d_2(v)$, $d_3(w)$ on intervals $[0,1]$, $[0,1]$, $[0,2]$, respectively.

Figure 3.26 shows an example with $d_1(u)$, $d_2(v)$ being cubic polynomial functions, and $d_3(w)$ being a linear function. This is a translational surface. Note that we have two

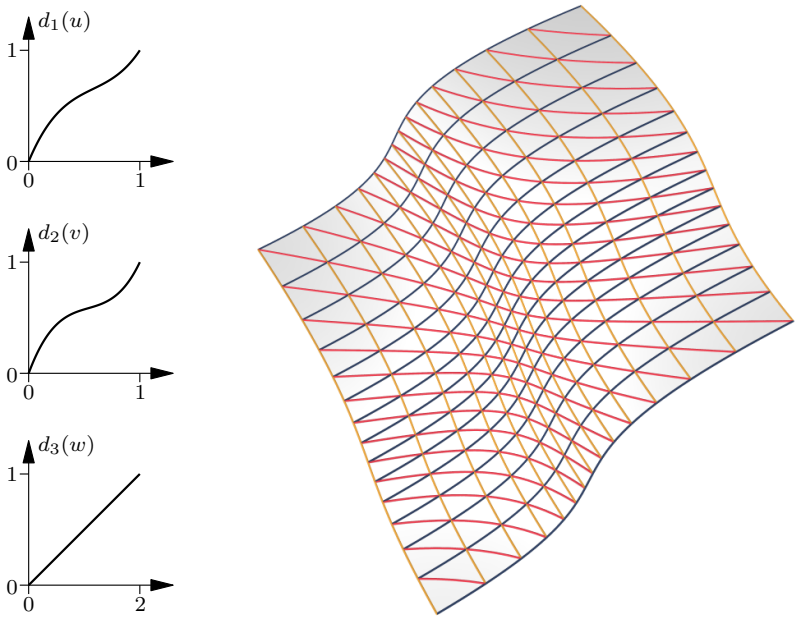


Figure 3.26: A continuous planar web (3.33) with $d_3(w)$ being a linear function, which is a translational surface.

parametric curves

$$c_1(u) = S(u, 0) = (d_1(u), d_2(0), d_3(c - u)), \qquad u \in [0, 1],$$

and

$$c_2(v) = S(0, v) = (d_1(0), d_2(v), d_3(c - v)), \qquad v \in [0, 1].$$

Since d_3 is a linear function, we have

$$d_3(c - u) + d_3(c - v) = d_3(c) + d_3(c - u - v).$$

From this

$$\begin{aligned} c_1(u) + c_2(v) &= (d_1(0) + d_1(u), d_2(0) + d_2(v), d_3(c) + d_3(c - u - v)) \\ &= S(u, v) + (d_1(0), d_2(0), d_3(c)). \end{aligned}$$

Therefore, $S(u, v)$ can be written as

$$S(u, v) = c_1(u) + c_2(v) - (d_1(0), d_2(0), d_3(c)),$$

which means that $S(u, v)$ is a translational surface [Farin 2001]. So we have: *The continuous planar web defined in Equation* (3.33) *is a translational surface if* d_3 *is a linear function.*

Note that the surface parametrization in (3.33) is singular at a point if the partial derivatives S_u and S_v at that point is linear dependent, i.e.,

$$S_u \times S_v = \mathbf{0}. \tag{3.34}$$

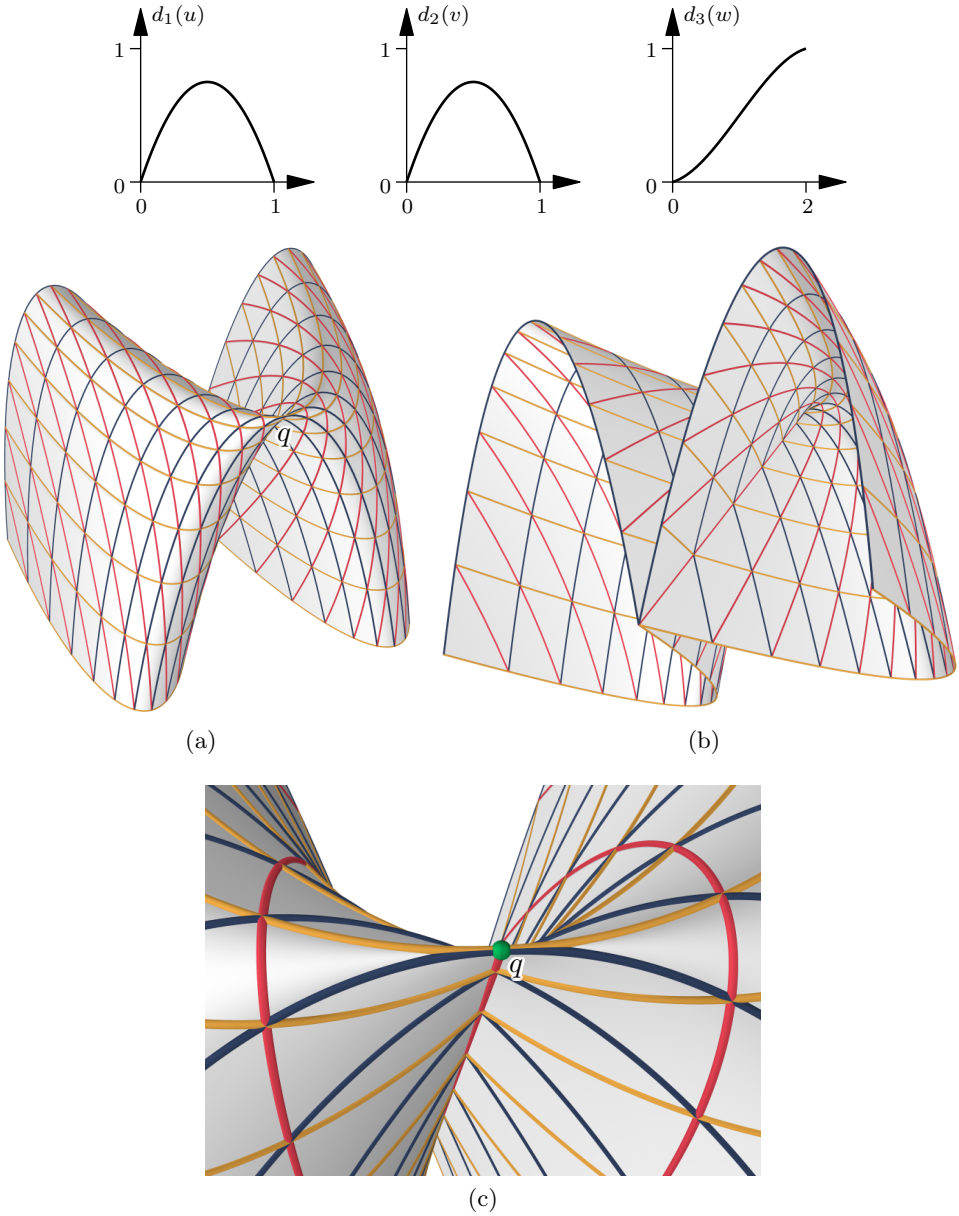
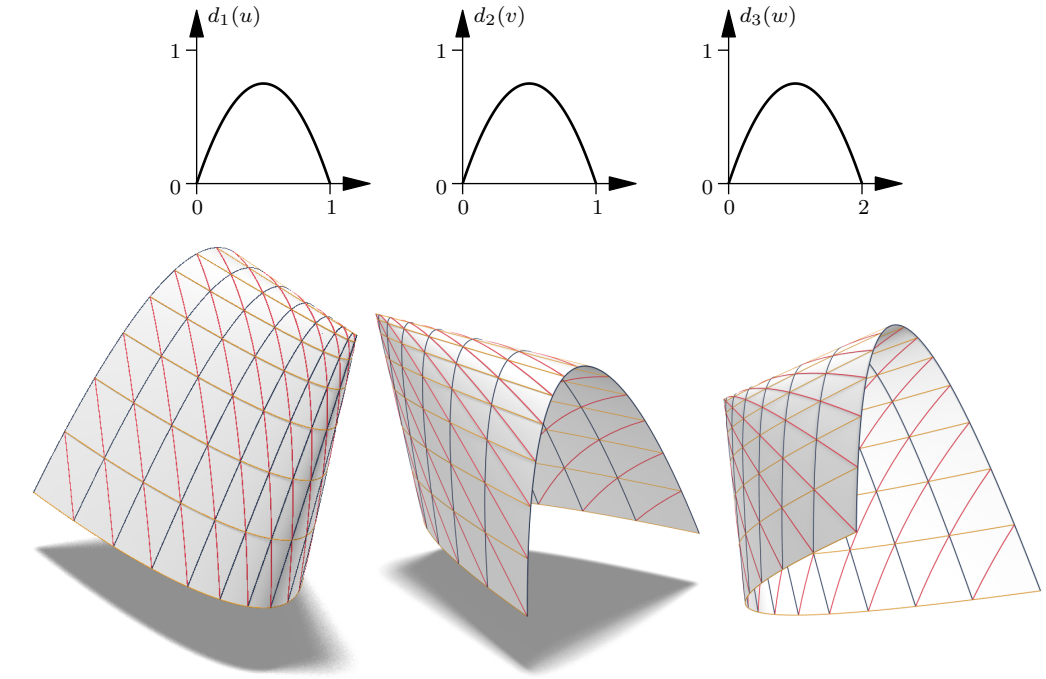


Figure 3.27: A continuous planar web with a singular point q. (a) The surface viewed from different angles. (b) A closer look at the singular point q (in green).

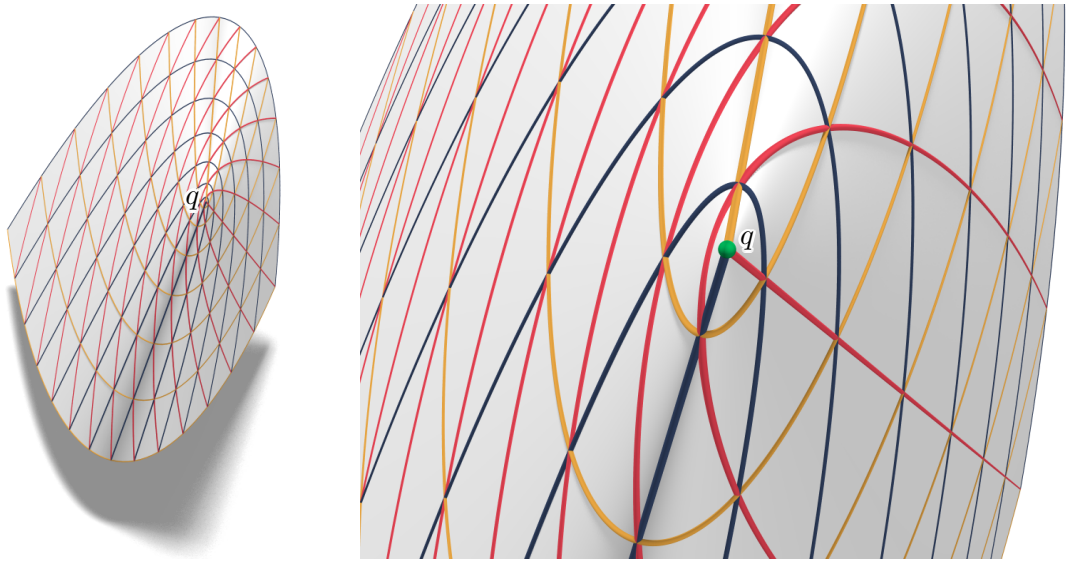
Since

$$S_u \times S_v = (d_2'(v)d_3'(c-u-v), d_1'(u)d_3'(c-u-v), d_1'(u)d_2'(v)),$$

condition (3.34) is satisfied if for the three derivative values $d_1'(u)$, $d_2'(v)$ and $d_3'(c-u-v)$, at least two of them are zero. Therefore, a singular point can be induced by stationary points of functions d_1, d_2, d_3 in the considered intervals. Figures 3.27 and 3.28 show surfaces constructed from functions d_1, d_2, d_3 with stationary points, as well as the



(a)



(b)

Figure 3.28: A continuous planar web with a singular point. (a) The surface viewed from different angles. (b) The singular point q (in green).

singular points. At the singular point in Figure 3.27, S_u, S_v are non-zero and parallel. At the singular point in Figure 3.28, $S_u = S_v = \mathbf{0}$. Also note that in Figure 3.27 there is self-intersection of the surface, while in Figure 3.28 every point on the surface is a double point.

Normal Planes of Parametric Curves

For a continuous one-parameter family $P(t)$ of planes, there exists a parametric curve $c(t)$ whose normal plane at point $c(t)$ coincides with $P(t)$, apart from the degenerate cases. If $P(t)$ is given, $c(t)$ can be found by solving an ODE. Therefore, we have

Proposition 3.1. *A continuous planar web is defined by the families of normal planes of $P_1(u)$, $P_2(v)$, $P_3(w)$ of three parametric curves $c_1(u)$, $c_2(v)$, $c_3(w)$ via Equation* (3.30).

Note that for a given family of planes $P(t)$, there may be more than one parametric curve whose normal planes are $P(t)$. For example, the following two curves

$$\begin{aligned} b_1(t) &= (R_1 \cos t, R_1 \sin t, 0), \\ b_2(t) &= (R_2 \cos t, R_2 \sin t, 0) \end{aligned}$$

are different if $R_1, R_2 > 0$ and $R_1 \neq R_2$, but they have the same family of normal planes. For the description of a continuous planar web, we can use any of the curves which give the desired family of normal planes.

In (3.32), the point $S(u, v)$ is at infinity if $\det(n_1(u), n_2(v), n_3(c-u-v)) = 0$ [Pottmann and Wallner 2001]. Such a point cannot be realized in an Euclidean space, and should be avoided in the construction of a continuous planar web. The surface $S(u, v)$ does not have points at infinity in a parameter domain R, if

$$\det(n_1(u), n_2(v), n_3(c - u - v)) \neq 0, \quad \text{for all } (u, v) \in R. \tag{3.35}$$

Figure 3.29 shows some continuous planar webs derived from the normal planes of three parametric curves. Condition (3.35) is used to guarantee that the surfaces are well defined in the considered parameter domains.

To control the shapes of continuous planar webs derived from normal planes, we need to control the parametric curves $c_1(u)$, $c_2(v)$, $c_3(w)$. In geometric modeling, B-spline curves are among the most widely used forms of parametric curves. Their shapes can be easily manipulated via their control points [Farin 2001]. In the following presentations, we will focus on the case where $c_1(u)$, $c_2(v)$, $c_3(w)$ are B-spline curves. Suppose the considered surface region is derived from normal planes of curve $c_i(t)$ on interval $t \in [a_i, b_i]$ $(i = 1, 2, 3)$. And we assume that $c_i(t)$ is a B-spline curve of degree n_i defined on a knot vector $T = \{t_0, \ldots, t_{m_i+n_i+1}\}$, where $t_{n_i} = a_i$, $t_{m_i+1} = b_i$, with control points $\{p_j^{(i)}\}_{j=0,\ldots,m_i}$, i.e.,

$$c_i(t) = \sum_{j=0}^{m_i} N_j^{n_i}(t) p_j^{(i)}. \tag{3.36}$$

Here $\{N_j^{n_i}(t)\}$ are B-spline basis functions of degree n_i defined on knot vector T.

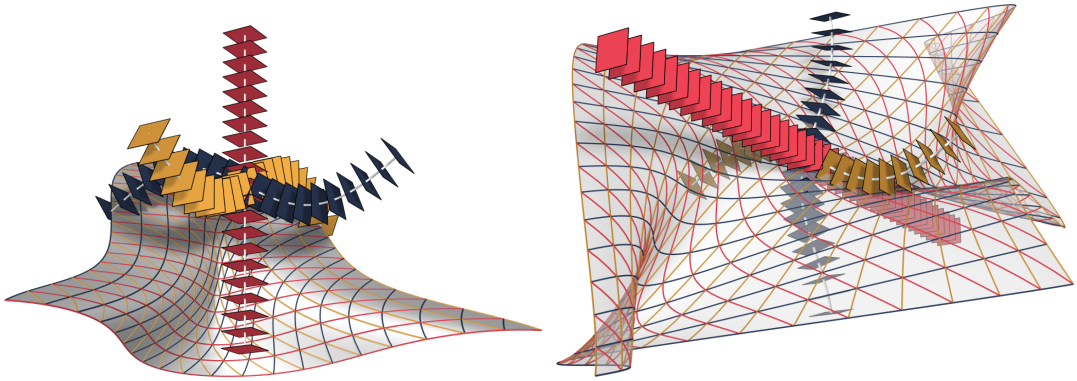


Figure 3.29: Continuous planar webs from normal planes of three curves. The normal planes containing the planar curves on the surface are shown in the same colors.

To ensure that condition (3.35) is satisfied on the considered parameter domain, we note that for a parametric curve $c(t)$, the normal vector of its normal plane is parallel to its tangent vector. If $c_1(u)$, $c_2(v)$, $c_3(w)$ are all regular curves, condition (3.35) is equivalent to

$$\det(\dot{c}_1(u), \dot{c}_2(v), \dot{c}_3(c-u-v)) \neq 0, \qquad \text{for all } (u,v) \in R, \tag{3.37}$$

where $\dot{c}_1$, $\dot{c}_2$, $\dot{c}_3$ are the first derivatives of $c_1(u)$, $c_2(v)$, $c_3(w)$, respectively. For the B-spline curve $c_i(t)$ defined in (3.36),

$$\dot{c}_i(t) = \sum_{j=0}^{m_i-1} \frac{n_i}{t_{j+n_i+1} - t_j} N_j^{n_i-1}(t) \Delta p_j^{(i)}, \tag{3.38}$$

where $\{N_j^{n_i-1}(s)\}$ are B-spline basis functions of degree $n_i - 1$ defined on a knot vector $\widetilde{T} = \{t_1, \ldots, t_{m_i+n_i}\}$, and $\Delta p_j^{(i)} = p_{j+1}^{(i)} - p_j^{(i)}$ are the difference vectors between neighboring control points. From this we obtain a sufficient condition for $S(u,v)$ to have no point at infinity in the considered parameter domain

Proposition 3.2. *If c_1, c_2, c_3 are B-spline curves defined in* (3.36)*, then the continuous planar web defined by the normal planes of c_1, c_2, c_3 has no point at infinity if*

$$\det(\Delta p_i^{(1)}, \Delta p_j^{(2)}, \Delta p_k^{(3)}) > 0, \quad \textit{for all } i,j,k, \tag{3.39}$$

or

$$\det(\Delta p_i^{(1)}, \Delta p_j^{(2)}, \Delta p_k^{(3)}) < 0, \quad \textit{for all } i,j,k. \tag{3.40}$$

Proof. The derivative in (3.38) can be written as

$$\dot{c}_i(t) = \sum_{j=0}^{m_i-1} B_j^{(i)}(t) \Delta p_j^{(i)} \tag{3.41}$$

with non-negative functions $\{B_j^{(i)}(t)\}$, such that for any t in the considered domain, not all of $\{B_j^{(i)}(t)\}$ are zero. Then in (3.37), the determinant expands to

$$\det(\dot{c}_1(u), \dot{c}_2(v), \dot{c}_3(w)) = \sum_{i,j,k} B_i^{(1)}(u) B_j^{(2)}(v) B_k^{(3)}(w) \det(\Delta p_i^{(1)}, \Delta p_j^{(2)}, \Delta p_k^{(3)}),$$

where $w = c - u - v$. Here the coefficient values $\{B_i^{(1)}(u) B_j^{(2)}(v) B_k^{(3)}(w)\}$ are non-negative, and not all of them are zero. Therefore, if condition (3.39) or (3.40) is satisfied, then $\det(\dot{c}_1(u), \dot{c}_2(v), \dot{c}_3(w))$ is always positive or always negative on the considered domain, which satisfies the condition (3.37) for no point at infinity. □

Note that among the functions $\{B_j^{(i)}(t)\}$ in (3.41) at most n_i of them are non-zero at a given parameter t, due to the local control property of B-spline curves. Therefore, for $\det(\dot{c}_1(u), \dot{c}_2(v), \dot{c}_3(w))$ to be non-zero, we only need to impose the condition $\det(\Delta p_i^{(1)}, \Delta p_j^{(2)}, \Delta p_k^{(3)}) > 0$ or $\det(\Delta p_i^{(1)}, \Delta p_j^{(2)}, \Delta p_k^{(3)}) < 0$ on those index triples (i, j, k) for which $B_i^{(1)}(u) B_j^{(2)}(v) B_k^{(3)}(w)$ is non-zero for a certain parameter (u, v) in the considered domain. We can find such a set $\mathcal{I}$ of effective triples as follows. Let the knot vectors of $c_1(u)$, $c_2(u)$, $c_3(w)$ be denoted by $U = \{u_0, \ldots, u_{m_1+n_1+1}\}$, $V = \{v_0, \ldots, v_{m_2+n_2+1}\}$, $W = \{w_0, \ldots, w_{m_3+n_3+1}\}$, respectively. For each pair of knot spans $[u_\alpha, u_{\alpha+1}]$ ($\alpha = n_1, \ldots, m_1$) and $[v_\beta, v_{\beta+1}]$ ($\beta = n_2, \ldots, m_2$) where $u_\alpha \neq u_{\alpha+1}$, $v_\beta \neq v_{\beta+1}$, the corresponding range of w is

$$w \in [c - u_{\alpha+1} - v_{\beta+1}, c - u_\alpha - v_\beta].$$

Let $[w_\gamma, w_\delta]$ ($\gamma, \delta \in \mathbb{Z}$) be the smallest knot span of W which contains the interval $[c - u_{\alpha+1} - v_{\beta+1}, c - u_\alpha - v_\beta]$. Then a triple (i, j, k) with $\alpha - n_1 \leq i \leq \alpha - 1$, $\beta - n_2 \leq j \leq \beta - 1$, $\gamma - n_3 \leq k \leq \delta - 2$ is contained in $\mathcal{I}$. To summarize, we have

Corollary 3.3. *The surface region considered in Proposition 3.2 has no point at infinity if*

$$\det(\Delta p_i^{(1)}, \Delta p_j^{(2)}, \Delta p_k^{(3)}) > 0, \quad \textit{for all } (i, j, k) \in \mathcal{I}, \tag{3.42}$$

or

$$\det(\Delta p_i^{(1)}, \Delta p_j^{(2)}, \Delta p_k^{(3)}) < 0, \quad \textit{for all } (i, j, k) \in \mathcal{I}. \tag{3.43}$$

3.3.4 Modification of Continuous Planar Webs

We can modify a continuous planar web into another one. Here we introduce two approaches to do so. These modification methods enable us to create new designs of continuous planar webs from existing ones, instead of starting from scratch using the methods in Section 3.3.3.

Modification by Projective Transformations

In a three-dimensional real projective space RP^3 (obtained by adding points at infinity to $\mathbb{R}^3$), a projective transformation is a mapping between points. Such a transformation ϕ is represented by a 4×4 regular matrix $\mathbf{M}$. A point with homogeneous coordinates $\mathbf{Q}$ (written as a column vector) is mapped by ϕ to a point with homogeneous coordinates $\mathbf{MQ}$. Such a mapping induce a mapping ψ between planes, represented by the 4×4 regular matrix $\mathbf{M}^{-T}$, such that a plane with homogeneous coordinates $\mathbf{P}$ is mapped by ψ to a plane with homogeneous coordinates $\mathbf{M}^{-T}\mathbf{P}$. If a point q is incident with a plane P, then point $\phi(q)$ is incident with plane $\psi(P)$ [Pottmann and Wallner 2001]. Therefore, if a point q is the intersection between three planes P_1, P_2, P_3, then point $\phi(q)$ is the intersection between three planes $\psi(P_1)$, $\psi(P_2)$, $\psi(P_3)$. For a continuous planar web $S(u, v) = P_1(u) \cap P_2(v) \cap P_3(w)$ generated by three families of planes $P_1(u)$, $P_2(v)$, $P_3(w)$, the three plane families can be mapped by ψ to three new families of planes $\overline{P}_1(u) = \psi(P_1(u))$, $\overline{P}_2(v) = \psi(P_2(v))$, $\overline{P}_3(w) = \psi(P_3(w))$. $\overline{P}_1(u)$, $\overline{P}_2(v)$, $\overline{P}_3(w)$ can be used to create a continuous planar web

$$\overline{S}(u, v) = \overline{P}_1(u) \cap \overline{P}_2(v) \cap \overline{P}_3(w),$$

and point $\overline{S}(u, v)$ is the image of $S(u, v)$ via mapping ϕ

$$\overline{S}(u, v) = \phi(S(u, v)).$$

Therefore, we can modify an existing planar web $S(u, v)$ by specifying a projective transformation ϕ and applying it to $S(u, v)$. A projective transformation is determined by two *fundamental sets* of points $\{q_i\}_{i=0,\dots,4}$ and $\{r_i\}_{i=0,\dots,4}$, which means that no four points in $\{q_i\}$ are coplanar, and no four points in $\{r_i\}$ are coplanar. The matrix $\mathbf{M}$ of the projective transformation is determined by conditions

$$\mathbf{MQ}_i = \lambda_i \mathbf{R}_i, \text{ for some } \lambda_i \in \mathbb{R} \setminus \{0\}, \quad i = 0, \dots, 4, \tag{3.44}$$

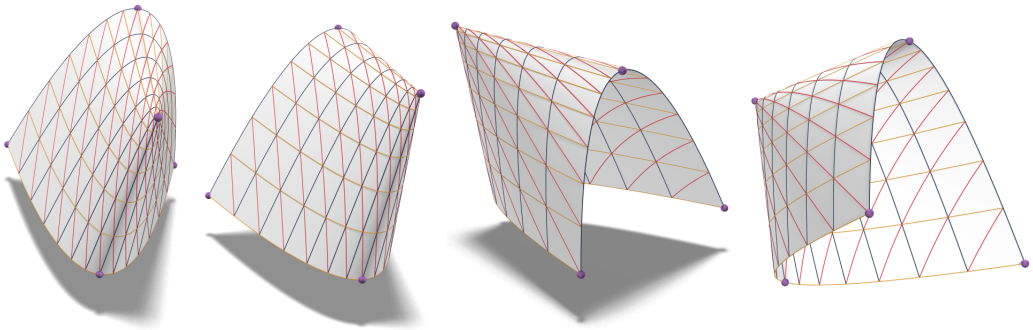
where $\mathbf{Q}_i$ and $\mathbf{R}_i$ are homogeneous coordinates of q_i and r_i, respectively [Pottmann and Wallner 2001]. Note that since $\{q_i\}$ is a fundamental set, vectors $\{\mathbf{Q}_i\}_{i=0,\dots,3}$ form a basis of $\mathbb{R}^4$, and $\mathbf{Q}_4$ can be represented as $\mathbf{Q}_4 = \sum_{i=0}^{3} a_i \mathbf{Q}_i$. Similarly, $\mathbf{R}_4$ can be represented as $\mathbf{R}_4 = \sum_{i=0}^{3} b_i \mathbf{R}_i$. Now denote $\widehat{\mathbf{Q}}_i = a_i \mathbf{Q}_i$, $\widehat{\mathbf{R}}_i = b_i \mathbf{R}_i$, $i = 0, \dots, 3$, then the condition (3.44) is equivalent to

$$\mathbf{M}\hat{\mathbf{Q}}_i = \hat{\mathbf{R}}_i, \quad i = 0, \dots, 3.$$

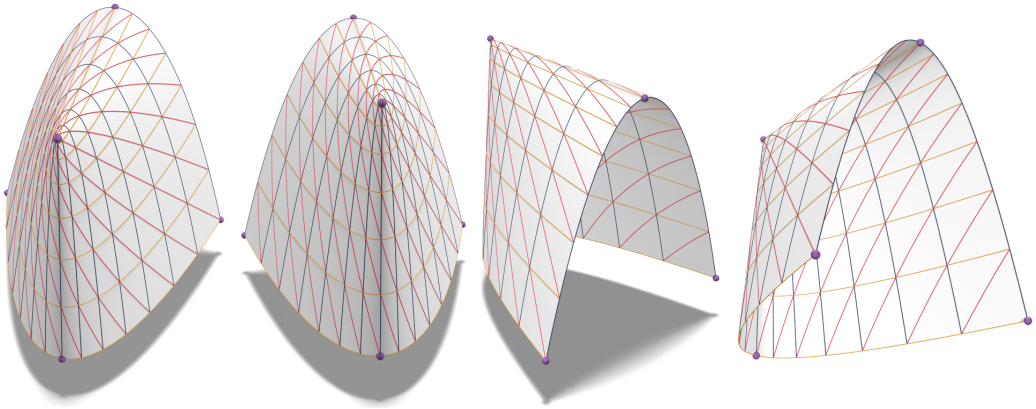
Therefore, $\mathbf{M}$ can be obtained by solving a linear system

$$\left[\widehat{\mathbf{Q}}_0 \widehat{\mathbf{Q}}_1 \widehat{\mathbf{Q}}_2 \widehat{\mathbf{Q}}_3\right]^T \mathbf{M}^T = \left[\widehat{\mathbf{R}}_0 \widehat{\mathbf{R}}_1 \widehat{\mathbf{R}}_2 \widehat{\mathbf{R}}_3\right]^T$$

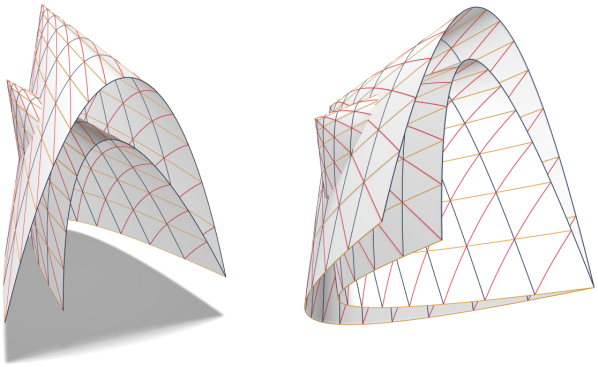
For the modification of a planar web $S(u, v)$ we can choose five points on $S(u, v)$ which form the source fundamental set $\{q_i\}$, and specify another five points as the image fundamental set $\{r_i\}$, determining the matrix $\mathbf{M}$. The new planar web $\overline{S}(u, v)$ will be incident with the points $\{r_i\}$. Figure 3.30 shows an example of such modification.



(a)



(b)



(c)

Figure 3.30: Modification of a continuous planar web using a projective transformation. (a) The original surface, with the source fundamental set of points (in purple), viewed from different angles. (b) The modified surface, with the target fundamental set of points (in purple), viewed from same angles as (a). (c) The original and modified surfaces are put together for comparison.

Note that under a projective transformation $\mathbf{M}$, some points on the original surface may be mapped to points at infinity. Let $\mathbf{P}_{\text{inf}}$ be the homogeneous coordinates of the plane at infinity, Then the plane P_0 with homogeneous coordinates $\mathbf{M}^T\mathbf{P}_{\text{inf}}$ is mapped to the plane at infinity. If P_0 intersects with the original surface, the intersection points will be mapped to points at infinity. To avoid points at infinity on the modified surface, we need to make sure that plane P_0 does not intersection with the original surface.

Interactive Deformation by Optimization

The second approach is only applicable to continuous planar webs derived from the normal planes of three B-spline curves. Starting from such a continuous planar web $S(u, v)$ defined on a parameter domain R, we sample a set of points $\{(u_i, v_i)\}$ inside R. The current surface points $\{S(u_i, v_i)\}$ at these parameter values give a set of "handles", which the user can drag to desired locations $\{q_i\}$. Then we search for a new continuous planar web $\overline{S}(u, v)$ by changing the control point positions of the three B-spline curves, such that points $\{\overline{S}(u_i, v_i)\}$ are close to $\{q_i\}$, This is done by minimizing a target functional

$$F = \lambda_{\text{fair}} f_{\text{fair}} + \sum_i w_i \|\overline{S}(u_i, v_i) - q_i\|^2, \tag{3.45}$$

where $w_i > 0$ are weights for each point, f_{fair} is a fairing functional for $\overline{S}(u, v)$, and λ_{fair} is the weight for f_{fair}. The variables are the control point coordinates of the three B-spline curves.

The weights $\{w_i\}$ in Equation (3.45) are chosen as follows. We separate all sample parameters $\{(u_i, v_i)\}$ into two sets: the "moved set" $\mathcal{M} = \{i \mid S(u_i, v_i) \neq q_i\}$ for which the target point location differ from the current surface point, and the "fixed set" $\mathcal{F} = \{i \mid S(u_i, v_i) = q_i\}$ for which the target point is the same as the current surface point. On the one hand, we want to make the new positions $\{\overline{S}(u_i, v_i)\}_{i\in\mathcal{I}}$ of the moved set close to their target positions. On the other hand, we want to achieve the effect that the surface region further away from the moved set is more constrained to its current position. We set $w_i = 1$ for each point in the moved set. The weight for a point in the fixed set depends on its distance $\mathcal{D}_i$ to the moved set. The larger this distance is, the larger the weight becomes. Let $\mathcal{D}_{\max} = \max_{i\in\mathcal{F}} \mathcal{D}_i$ and $\mathcal{D}_{\min} = \min_{i\in\mathcal{F}} \mathcal{D}_i$ be the maximum and minimum values of such distances. Then $w_i (i \in \mathcal{F})$ is defined as

$$w_i = \frac{\mathcal{D}_i - \mathcal{D}_{\min}}{\mathcal{D}_{\max} - \mathcal{D}_{\min}}\alpha + \frac{\mathcal{D}_{\max} - \mathcal{D}_i}{\mathcal{D}_{\max} - \mathcal{D}_{\min}}\beta,$$

where α, $\beta \in \mathbb{R}$ are user-defined values such that $0 \leq \beta \ll \alpha \leq 1$. The distance $\mathcal{D}_i$ is defined as the minimum distance to a point in the moved set, measured either in the parameter domain

$$\mathcal{D}_i = \min_{j\in\mathcal{M}} \|(u_i, v_i) - (u_j, v_j)\|, \tag{3.46}$$

or using the Euclidean distance between current surface points

$$\mathcal{D}_i = \min_{j\in\mathcal{M}} \|S(u_i, v_i) - S(u_j, v_j)\|. \tag{3.47}$$

To define the fairing term in (3.45), we take the intersection points between the three families of planar curves $C_1(i)$, $C_2(j)$, $C_3(k)$ defined in (3.31). These points are the vertices of a regular mesh of a discrete planar web, where the three families of mesh polylines $\mathscr{L}_1$, $\mathscr{L}_2$, $\mathscr{L}_3$ correspond to $C_1(i)$, $C_2(j)$, $C_3(k)$, respectively. We define the fairing term based on these polylines

$$f_{\text{fair}} = \sum_{i=1,2,3} f_{\text{fair},\sigma}(\mathscr{L}_i), \qquad \sigma = \text{II}, \text{III},$$

with $f_{\text{fair},\sigma}(\mathscr{L}_i)$ defined in Equation (3.4).

Since the initial surface $S(u, v)$ is constructed from three B-spline curves according to Section 3.3.3, the initial control points satisfy (3.42) or (3.43). To ensure that the new surface $\overline{S}(u, v)$ is free of points at infinity, we need to maintain a condition

$$\eta \cdot \det(\Delta p_i^{(1)}, \Delta p_j^{(2)}, \Delta p_k^{(3)}) > 0, \quad \text{for all } (i, j, k) \in \mathcal{I}, \tag{3.48}$$

where $\eta = 1$ if the initial surface $S(u, v)$ satisfies (3.42), and $\eta = -1$ if $S(u, v)$ satisfies (3.43). $\mathcal{I}$ is the same as in (3.42) and (3.43). In other words, we maintain the same sufficient condition for no point at infinity as the initial surface. Therefore, the new surface $\overline{S}(u, v)$ is obtained by minimizing the target function (3.45) subject constraints (3.48). This constrained optimization problem is solved by sequential quadratic programming [Nocedal and Wright 2006]. Our implementation uses the SLSQP algorithm implemented in NLopt library [Johnson 2011] which is based on [Kraft 1988] [Kraft 1994].

Figure 3.31 shows two examples of such modification. Figure 3.31(a) shows the original surfaces, as well as the B-spline curves and the normal planes. The handles are sampled as the intersection points of a network of planar curves (shown in red) defined in (3.31). Among the target positions of the handles (shown in white), only one of them is different from the original position (marked as q_1). Figures 3.31(b) and 3.31(c) are the modified surfaces, using weighting schemes defined by (3.47) and (3.46), respectively.

Table 3.2 shows some details of the optimization. Here N_{cp} is the total number of control points for the B-spline curves. #var is the number of variables for the optimization problem. N_{handle} is the number of handles. N_{fair} is the number of sample points for the fairing term. T is the timing in seconds.

Figure	N_{cp}	#var	N_{handle}	N_{fair}	T
3.31 Left	15	45	49	2401	6
3.31 Right	21	63	36	2304	3

Table 3.2: Details and timing for the deformation in Figure 3.31.

3.4 Discussion

There are limitations to the discrete web optimization, especially to finding webs such that both a reference surface is approximated and geometric properties are fulfilled.

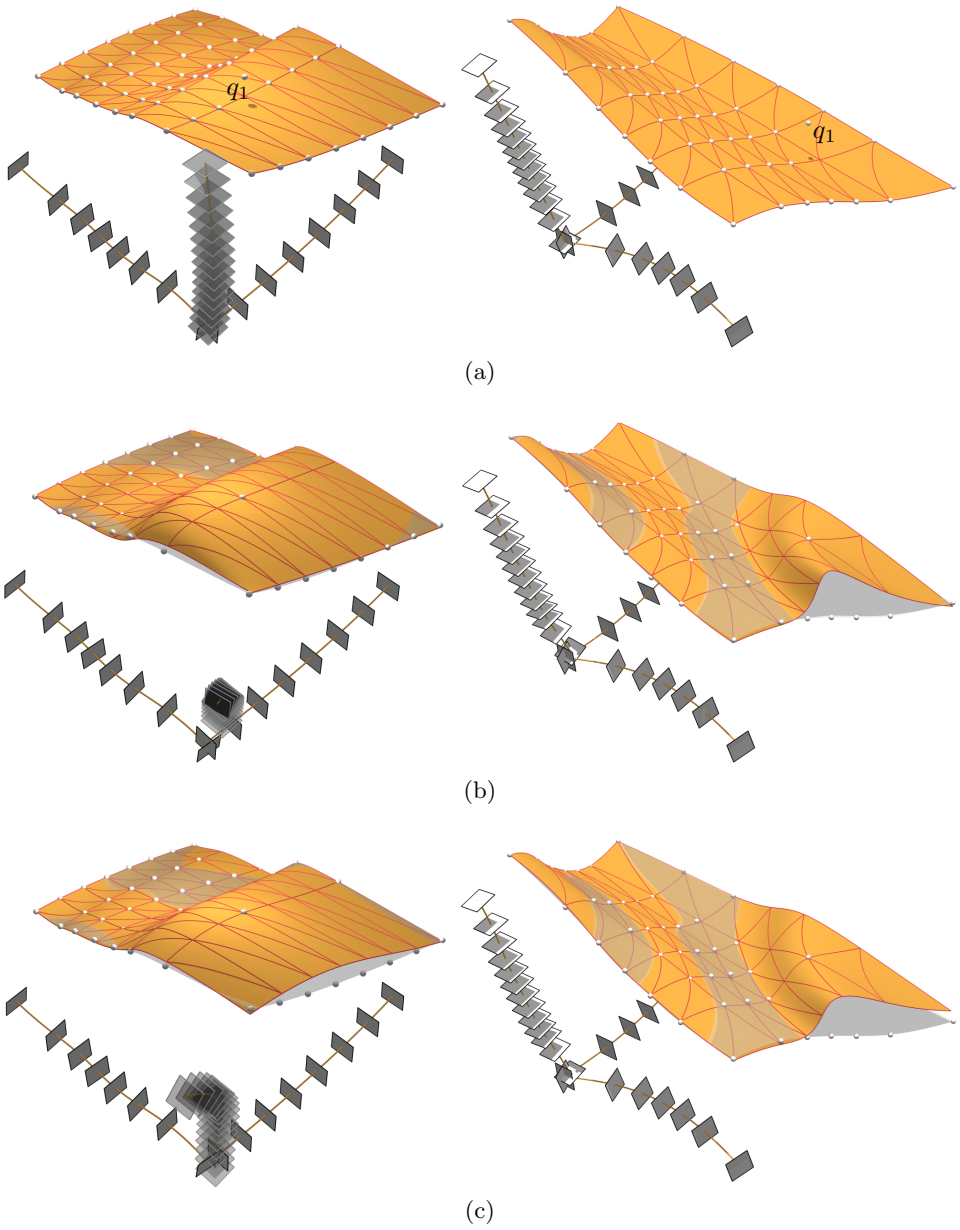


Figure 3.31: Interactive deformation of continuous planar webs. (a) The continuous planar webs before deformation, with one target point q_1 different from the original position of the handle. (b) Deformed surface using "Euclidean" weights (3.47). The original surface shape is also shown for comparison. (b) Deformed surface using "parametric" weights (3.46).

This is due to the global and coupled nature of constraints. For some surfaces there simply exists no web which can satisfy all the constraints without large deviation from the surface. For such cases our algorithm will fail to produce satisfactory results. It remains a challenging research problem to give a precise characterization of the types of webs that can exist on a given surface.

Due to the global nature of the constraints, local control of webs can be expected to be available only to a limited extent. For the case of nontrivial planar webs, any change of one of the families of planes will influence the surface at least along strips. For geodesic webs covering a given surface S_0, a dimension count in the smooth case shows that we cannot expect that locally deformed surfaces S_t $(0 \leq t < \varepsilon)$ are still covered by a geodesic web. By [Volk 1929], S_t must satisfy a certain 4th order PDE. If the deformation is local, all S_t's share values and derivatives along a common boundary. This usually implies that all S_t's are the same.

A combinatorial limitation is the regularity of meshes we use. It is possible to extend the discrete web optimization approach to irregular meshes (i.e., meshes containing interior vertices of valence $\neq 6$). For an interior vertex v of even valence which is different from 6 (a singular vertex), we can still categorize opposite neighboring vertices into the same mesh polyline containing v and apply the functionals in Section 3. However, this does not make sense in most cases. For instance, requiring the geodesic property near a singular vertex leads to a singularity of the surface geometry. This implication "combinatorial singularity $\Longrightarrow$ geometric singularity" follows from the existence of geodesic mesh polylines which run side by side before encountering the singularity and which diverge immediately afterwards, implying a concentration of Gaussian curvature (see the Jacobi equation (2.2)). Other work on regular triangular and hexagonal parametrizations such as [Nieser et al. 2010] does not surfer from this problem.

4 Conclusion and Future Work

In this thesis, we described methods to create patterns of special curves on surfaces, which find applications in design and realization of freeform architecture.

We first presented an evolution approach to generate a series of curves which are either geodesic or piecewise geodesic, starting from a given source curve on a surface. This approach "propagates" curves across the surface so that the distance functions between neighboring curves are close to given target functions. Each new curve is computed according to a Jacobi field on the current "wavefront" curve of the propagation which is closest to the target distance function. The Jacobi fields give first order approximations of distance fields from the wavefront curve to a neighboring curve. The problem of selecting a Jacobi field that is closest to the target distance function is easily solvable, thanks to the linear structure of the space of Jacobi fields. The use of target distance functions provides flexibility in the creation of curve patterns. In the simplest case of constant target distance functions, we obtain curve patterns which represent layouts of panels with straight developments and approximately constant widths, which are important for fabrication of architectural freeform shapes and for interior design. Using other types of target distance functions, we are able to obtain non-trivial patterns adapted to specific needs for design and construction. Using this evolution approach, we can explore different patterns by specifying different source curves and target distance functions. This leads to more predictable and controllable results than the nonlinear optimization approach to geodesic D-strip models in [Pottmann et al. 2008b].

We then presented computational tools to create functional webs, i.e., three families of curves with regular connectivity, where the curves have designated special properties (planar, circular or geodesic). By discretizing a web as a regular triangle mesh, we can compute a discrete web using an optimization approach, where the target functional penalizes the deviation of the curves from their target properties. To investigate the solvability of this optimization problem for specific target properties, we estimate the number of d.o.f for a functional web as the difference between the number of mesh vertex coordinates and the number of constraints induced by curve properties. This estimation helps the user to choose target curve properties which are achievable for a specific web. For webs consisting of planar curves, we show how they can be computed in an exact way using three families of planes. To the best of our knowledge, this is the first systematic investigation into practical computation of functional webs. Recent emergence of web structures in architecture poses new challenges in their realization, since every general-shaped curve element has to be fabricated by CNC-machining, resulting in high manufacturing cost [Scheurer 2010]. Our computational framework can help the architects to design webs with curve elements that are easy to fabricate, leading to economical realization of such structures.

For future work on geodesic patterns, we would like to extend the evolution approach

to address some current limitation. The current approach of evolution only works for surface patches of disk topology. In order to use this approach to cover a surface with other topologies, we need to segment the surface into patches, e.g., using the segmentation algorithm from [Pottmann et al. 2010]. Such segmentation may not always be desirable. For example, for a surface of revolution, we would like the generated pattern to maintain symmetry with respect to the rotation axis. However, the segmentation approach cannot guarantee such symmetry. Instead, we can apply the evolution approach to closed curves which are piecewise geodesic. The Jacobi fields on such piecewise geodesic closed curves can be computed in a way similar to the Jacobi fields on piecewise geodesic open curves. Such closed curves can be propagated on a surface of cylinder topology without the need for segmentation. Besides, when segmenting a complicated surface for geodesic patterns, the user may want to control the positions of singularities, i.e., points where more than two patches meet. We can work on a segmentation algorithm which respects specified singularity positions, and produces patches of disk or cylinder topology which are suitable for the evolution of geodesic/piecewise geodesic curves.

For future work on functional webs, an interesting topic is the exploration of shape spaces of functional webs with given curve properties. This can be done following the approach of [Yang et al. 2011]: A web discretized as a regular triangle mesh can be represented by a vector in $\mathbb{R}^{3|V|}$ which includes the mesh vertex coordinates, where $|V|$ is the number of vertices. A target property of an edge polyline defines a set of constraints for the coordinates of its vertices. Therefore, for a given set of curve properties, the webs with these properties correspond to a manifold in $\mathbb{R}^{3|V|}$ defined by the corresponding constraints. If this manifold is not an empty set of points, we can allow the user to explore it to see the possible shapes of webs. Such exploration is useful in the design of a web. Another interesting and related topic is the existence of discrete webs with given curve properties. The estimation of the number of d.o.f. for a web in Chapter 3 is only a rough one. First of all, the constraints derived from curve properties may not be independent, or may contradict each other. These cases are not taken into account by the current estimation. For example, according to this estimation, we cannot construct a web where all curves are circular, because the number of constraints exceeds the number of vertex coordinates. However, as shown in [Pottmann et al. 2011], such webs do exist. Moreover, the constraints from curve properties are all nonlinear. Therefore, the difference between the number of variables and the number of constraints may not reflect the actual degrees of freedom for the solutions. The existence of discrete webs with given properties is a challenging yet important topic. Its solution will shed light on the problem of existence of hexagonal webs with corresponding properties, which remains an open topic in web geometry.

Within architectural geometry, an important future work is *construction-aware design* for freeform architecture. Traditionally in an architectural project, design and rationalization are separate tasks. The architect first uses CAD tools to create the designed shape. This design then goes through the rationalization process, which generates a simplified model that is suitable for fabrication. The rationalization process usually involves complicated nonlinear optimization, and the output model may not always be consistent with the architect's original intent. In that case, the architect needs to modify the

original design, and apply rationalization again. Sometimes this cycle between design and rationalization needs to be iterated for a few times before a satisfactory result is obtained. Such iteration is time-consuming. Also, it is not easy to know what changes on the original design are needed to produce a specific rationalized shape. Therefore, researchers have recently started working on construction-aware design tools, which integrate rationalization into the design process. Such tools will generate only those designs which meet the specified manufacturing and feasibility constraints. For the geodesic pattern problem, we may start from some target distance functions, and look for surfaces which allow families of geodesic/piecewise geodesic curves with distance functions between neighboring curves matching the target functions. For functional webs, we can start from a set of target curve properties, and look for webs consisting of curves with these properties. In both cases, the feasible shapes may form a space, and the idea of shape space exploration will be useful in the design process.

Bibliography

Aleksandrov, A. D., Zahlgaller, V. A., 1967. Intrinsic Geometry of surfaces. American Mathematical Society.

Alliez, P., de Verdière, É. C., Devillers, O., Isenburg, M., 2003. Isotropic surface remeshing. In: Proceedings of Shape Modeling International. pp. 49–58.

Bentley, J. L., 1975. Multidimensional binary search trees used for associative searching. Communications of the ACM 18 (9), 509–517.

Blaschke, W., Bol, G., 1938. Geometrie der Gewebe. Springer.

Bo, P., Pottmann, H., Kilian, M., Wang, W., Wallner, J., 2011. Circular arc structures. ACM Transactions on Graphics 30 (4), 101:1–101:12, Proceedings of SIGGRAPH 2011.

Cazals, F., Pouget, M., 2008. Algorithm 889: Jet_fitting_3: —A generic C++ package for estimating the differential properties on sampled surfaces via polynomial fitting. ACM Transactions on Mathematical Software 35 (3), 24:1–24:20.

Ceccato, C., Hesselgren, P., Pauly, M., Pottmann, H., Wallner, J. (Eds.), 2010. Advances in Architectural Geometry 2010. Springer.

Chen, J., Han, Y., 1990. Shortest paths on a polyhedron. In: Proceedings of the sixth annual symposium on Computational geometry. pp. 360–369.

Chen, Y., Davis, T. A., Hager, W. W., Rajamanickam, S., 2008. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. ACM Transactions on Mathematical Software 35 (3), 22:1–22:14.

Chern, S.-S., 1982. Web geometry. Bulletin of the American Mathematical Society 6, 1–8.

Chernov, N., 2010. Circular and Linear Regression: Fitting Circles and Lines by Least Squares. CRC Press.

de Berg, M., Cheong, O., van Kreveld, M., Overmars, M., 2008. Computational Geometry: Algorithms and Applications, 3rd Edition. Springer.

do Carmo, M. P., 1976. Differential Geometry of Curves and Surfaces. Prentice Hall.

do Carmo, M. P., 1992. Riemannian Geometry. Birkhäuser.

Eigensatz, M., Kilian, M., Schiftner, A., Mitra, N. J., Pottmann, H., Pauly, M., 2010. Paneling architectural freeform surfaces. ACM Transactions on Graphics 29, 45:1–45:10, Proceedings of SIGGRAPH 2010.

Farin, G., 2001. Curves and Surfaces for CAGD: A Practical Guide, 5th Edition. Morgan Kaufmann.

Flöry, S., Pottmann, H., 2010. Ruled surfaces for rationalization and design in architecture. In: Sprecher, A., Yeshayahu, S., Lorenzo-Eiroa, P. (Eds.), LIFE in:formation. On Responsive Information and Variations in Architecture. Association for Computer Aided Design in Architecture (ACADIA), pp. 103–109, Proceedings of ACADIA 2010.

Fu, C.-W., Lai, C.-F., He, Y., Cohen-Or, D., 2010. K-set tilable surfaces. ACM Transactions on Graphics 29 (4), 44:1–44:6, Proceedings of SIGGRAPH 2010.

Glymph, J., Shelden, D., Ceccato, C., Mussel, J., Schober, H., 2004. A parametric strategy for free-form glass structures using quadrilateral planar facets. Automation in Construction 13 (2), 187–202, Conference of the Association for Computer Aided Design in Architecture.

Graf, H., Sauer, R., 1924. Über dreifache Geradensysteme. Sitzungsberichte der Bayerische Akademie der Wissenschaften Mathematisch-Naturwissenschaftliche Abteilung, 119–156.

Johnson, S. G., 2011. The NLopt nonlinear-optimization package. `http://ab-initio.mit.edu/nlopt`.

Jolliffe, I. T., 2002. Principal Component Analysis, 2nd Edition. Springer.

Kahlert, J., Olson, M., Zhang, H., 2011. Width-bounded geodesic strips for surface tiling. The Visual Computer 27 (1), 45–56.

Kaneva, B., O'Rourke, J., 2000. An implementation of Chen & Han's shortest paths algorithm. In: Proceedings of the 12th Canadian Conference on Computational Geometry.

Kimmel, R., Sethian, J. A., 1998. Computing geodesic paths on manifolds. Proceedings of the National Academy of Sciences of the United States of America 95 (15), 8431–8435.

Kirsanov, D., 2004. Minimal discrete curves and surfaces. Ph.D. thesis, Harvard University.

Kraft, D., 1988. A software package for sequential quadratic programming. Tech. Rep. DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen.

Kraft, D., 1994. Algorithm 733: TOMP – Fortran modules for optimal control calculations. ACM Transactions on Mathematical Software 20 (3), 262–281.

Kuipers, J. B., 2002. Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality. Princeton University Press.

Lazareva, V. B., 1988. Parallelizable three-webs formed by pencils of circles. Webs and Quasi-groups, 74–77 (in Russian).

Lindsey, B., 2001. Digital Gehry: Material Resistance, Digital Construction. Birkhäuser.

Liu, Y., Pottmann, H., Wallner, J., Yang, Y.-L., Wang, W., 2006. Geometric modeling with conical meshes and developable surfaces. ACM Transactions on Graphics 25 (3), 681–689, Proceedings of SIGGRAPH 2006.

Liu, Y.-J., Zhou, Q.-Y., Hu, S.-M., 2007. Handling degenerate cases in exact geodesic computation on triangle meshes. The Visual Computer 23 (9), 661–668.

Madsen, K., Nielsen, H. B., Tingleff, O., 2004. Methods for non-linear least squares problems. `http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3215/pdf/imm3215.pdf`.

Martínez, D., Velho, L., Carvalho, P. C., 2005. Computing geodesics on triangular meshes. Computers & Graphics 29 (5), 667–675.

Mayrhofer, K., 1931. Über Sechseckgewebe aus Geodätischen. Monatshefte für Mathematik 38 (1), 401–404.

Mitchell, J. S. B., Mount, D. M., Papadimitriou, C. H., 1987. The discrete geodesic problem. SIAM Journal on Computing 16 (4), 647–668.

Mitchell, W. J., 2001. Roll over Euclid: how Frank Gehry designs and builds. In: Ragheb, J. F. (Ed.), Frank Gehry, Architect. Guggenheim Museum Publications, pp. 353–363.

Nieser, M., Palacios, J., Polthier, K., Zhang, E., 2010. Hexagonal global parameterization of arbitrary surfaces. In: ACM SIGGRAPH ASIA 2010 Sketches. pp. 5:1–5:2.

Nocedal, J., Wright, S., 2006. Numerical Optimization, 2nd Edition. Springer.

Novotni, M., Klein, R., 2002. Computing geodesic distances on triangular meshes. In: The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2002 (WSCG'2002).

Pereira, J. V., Pirio, L., 2009. An invitation to web geomety — from abel's addition theorem to the algebraization of codimension one webs. `http://w3.impa.br/~jvp/invitation.pdf`.

Pirazzi, C., Weinand, Y., 2006. Geodesic lines on free-form surfaces – optimized grids for timber rib shells. In: Proceedings of World Conference in Timber Engineering.

Polthier, K., Schmies, M., 1998. Straightest geodesics on polyhedral surfaces. In: Hege, H.-C., Polthier, K. (Eds.), Mathematical Visualization. Springer Verlag, Heidelberg, pp. 135–150.

Pottmann, H., 2009. Geometry and new and future spatial patterns. Architectural Design 79 (6), 60–65.

Pottmann, H., 2010. Architectural geometry as design knowledge. Architectural Design 80 (4), 72–77.

Pottmann, H., Asperl, A., Hofer, M., Kilian, A., 2007a. Architectural Geometry. Bentley Institute Press.

Pottmann, H., Hofer, M., Kilian, A. (Eds.), 2008a. Advances in Architectural Geometry 2008.

Pottmann, H., Huang, Q., Deng, B., Schiftner, A., Kilian, M., Guibas, L., Wallner, J., 2010. Geodesic patterns. ACM Transactions on Graphics 29 (4), 43:1–43:10, Proceedings of SIGGRAPH 2010.

Pottmann, H., Leopoldseder, S., 2003. A concept for parametric surface fitting which avoids the parametrization problem. Computer Aided Geometric Design 20 (6), 343–362.

Pottmann, H., Liu, Y., Wallner, J., Bobenko, A., Wang, W., 2007b. Geometry of multilayer freeform structures for architecture. ACM Transactions on Graphics 26 (3), Proceedings of SIGGRAPH 2007.

Pottmann, H., Schiftner, A., Bo, P., Schmiedhofer, H., Wang, W., Baldassini, N., Wallner, J., 2008b. Freeform surfaces from single curved panels. ACM Transactions on Graphics 27 (3), 76:1–76:10, Proceedings of SIGGRAPH 2008.

Pottmann, H., Schiftner, A., Wallner, J., 2008c. Geometry of architectural freeform structures. Internationale Mathematische Nachrichten 209, 15–28.

Pottmann, H., Shi, L., Skopenkov, M., 2011. Darboux cyclides and webs from circles. ArXiv e-prints.

Pottmann, H., Wallner, J., 2001. Computational Line Geometry. Springer.

Pouget, M., Cazals, F., 2011. Estimation of local differential properties. In: CGAL User and Reference Manual, 3.8 Edition. CGAL Editorial Board, `http://www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg:Jet_fitting_3`.

Pratt, V., 1987. Direct least-squares fitting of algebraic surfaces. ACM SIGGRAPH Computer Graphics 21 (4), Proceedings of SIGGRAPH 1987.

Scheurer, F., 2010. Materialising complexity. Architectural Design 80 (4), 86–93.

Schiftner, A., Höbinger, M., Wallner, J., Pottmann, H., 2009. Packing circles and spheres on surfaces. ACM Transactions on Graphics 28, 139:1–139:8, Proceedings of SIGGRAPH Asia 2009.

Shelden, D. R., 2002. Digital surface representation and the constructibility of gehry's architecture. Ph.D. thesis, Massachusetts Institute of Technology.

Shelekhov, A., 2007. Classification of regular three-webs formed by pencils of circles. Journal of Mathematical Sciences 143 (6), 3607–3629.

Singh, M., Schaefer, S., 2010. Triangle surfaces with discrete equivalence classes. ACM Transactions on Graphics 29 (4), 46:1–46:7, Proceedings of SIGGRAPH 2010.

Spuybroek, L., 2004. NOX: Machining Architecture. Thames & Hudson.

Surazhsky, V., Surazhsky, T., Kirsanov, D., Gortler, S. J., Hoppe, H., 2005. Fast exact and approximate geodesics on meshes. ACM Transactions on Graphics 24 (3), 553–560, Proceedings of SIGGRAPH 2005.

Volk, O., 1929. Über Flächen mit geodätischen Dreiecksnetzen. Sitzungsberichte der Heidelberger Akademie der Wissenschaften.

Wallner, J., Pottmann, H., 2011. Geometric computing for freeform architecture. Journal of Mathematics in Industry 1 (1), 1–18.

Wallner, J., Pottmann, H., Hofer, M., 2007. Fair webs. The Visual Computer 23 (1), 83–94.

Wallner, J., Schiftner, A., Kilian, M., Flöry, S., Höbinger, M., Deng, B., Huang, Q., Pottmann, H., 2010. Tiling freeform shapes with straight panels: Algorithmic methods. In: Ceccato, C., et al. (Eds.), Advances in Architectural Geometry 2010. Springer, pp. 73–86.

Wang, W., Liu, Y., Yan, D.-M., Chan, B., Ling, R., Sun, F., 2008. Hexagonal meshes with planar faces. Tech. Rep. TR-2008-13, Department of Computer Science, The University of Hong Kong.

Xin, S.-Q., Wang, G.-J., 2009. Improving Chen and Han's algorithm on the discrete geodesic problem. ACM Transactions on Graphics 28 (4), 104:1–104:8.

Xin, S.-Q., Wang, G.-J., 2010. Applying the improved chen and han's algorithm to different versions of shortest path problems on a polyhedral surface. Computer-Aided Design 42 (10), 942–951.

Yang, Y.-L., Yang, Y.-J., Pottmann, H., Mitra, N. J., 2011. Shape space exploration of constrained meshes. In: SIGGRAPH Asia 2011.

Yatziv, L., Bartesaghi, A., Sapiro, G., 2006. O(N) implementation of the fast marching algorithm. Journal of Computational Physics 212 (2), 393–399.

Zadravec, M., Schiftner, A., Wallner, J., 2010. Designing quad-dominant meshes with planar faces. Computer Graphics Forum 29 (5), 1671–1679, Proceedings of Symposium on Geometry Processing 2010.

Curriculum Vitae

Bailin Deng, born August 21, 1982 in Xinhui, Guangdong Province, China.

Education

09/2001 - 06/2005	Tsinghua University, China. BEng in Computer Software.
09/2003 - 06/2004	The Chinese University of Hong Kong, Hong Kong SAR, China. Exchange student.
09/2005 - 06/2008	Tsinghua University, China. MSc in Computer Science.
since 10/2008	Vienna University of Technology, Austria. Ph.D. studies in Applied Geometry, supervisor Helmut Pottmann.

Employment and Research Experience

since 10/2008	Research assistant at Vienna University of Technology, Austria. Research in Applied Geometry.
11/2009 - 01/2010	Visiting student at Geometric Modeling and Scientific Visualization Center, King Abdullah University of Science and Technology, Saudi Arabia. Research in Architectural Geometry.

Publications

Deng, B., Pottmann, H., Wallner, J., 2011. Functional webs for freeform architecture. Computer Graphics Forum 30 (5), 1369–1378, Proc. Symp. Geometry Processing 2011.

Pottmann, H., Huang, Q., Deng, B., Schiftner, A., Kilian, M., Guibas, L., Wallner, J., 2010. Geodesic patterns. ACM Transactions on Graphics 29 (4), 43:1–43:10, Proc. SIGGRAPH 2010.

Wallner, J., Schiftner, A., Kilian, M., Flöry, S., Höbinger, M., Deng, B., Huang, Q., Pottmann, H., 2010. Tiling freeform shapes with straight panels: Algorithmic methods. In: Ceccato, C., et al. (Eds.), Advances in Architectural Geometry 2010. Springer, pp. 73–86.

Yong, J.-H., Deng, B., Cheng, F., Wang, B., Wu, K., Gu, H.-J., 2009. Removing local irregularities of triangular meshes with highlight line models. Science in China Series F: Information Sciences 52 (3), 418–430.

Talks

07/2011	*Functional Webs for Freeform Architecture*, Symposium on Geometry Processing 2011, Lausanne.